Eventus ID
Supervised Event Coding From Text Written in Spanish
Version 2.0

Javier Osorio
&
Alejandro Reyes

June, 2014

Legal Notice

Copyright

Terms of Use

How to Cite This Program

Please cite program as:

Javier Osorio and Alejandro Reyes. 2014. Eventus ID. Supervised Event Coding From Text Written in Spanish. Version 2.0

BibTex citation:

```
@misc{Osorio2014,
address = {Puebla, Puebla},
author = {Osorio, Javier and Reyes, Alejandro},
title = {{Eventus ID. Supervised Event Coding from
Text Written in Spanish}},
url = {http://www.javierosorio.net/#!software/cqbi},
year = {2014}
}
```

Report Bugs

Please report any bugs to: javier.osoriozago@gmail.com

Updated copies of the EVENTUS ID program and manual can be found at http://www.javierosorio.net/#!software/cqbi

Latest manual update: June 10, 2014

Acknowledgements

The development of EVENTUS ID was possible thanks to the generous financial support provided by the Doctoral Dissertation Research Improvement Grant of the National Science Foundation (award SES–1123572); the Drugs, Security and Democracy (DSD) Fellowship of the Social Science Research Council and the Open Society Foundations; the Jennings Randolph Peace Scholarship Dissertation Program of the United States Institute of Peace; and the Graduate Research Grant of the Kellogg Institute for International Studies at the University of Notre Dame.

Further refinements of the program were possible thanks to the opportunities, time and space provided by the Program on Order, Conflict and Violence at Yale University; the Dissertation Fellowship of the Harry Frank Guggenheim Foundation; and the Mario Einaudi Center for International Studies at Cornell University.

Special thanks to Philip Schrodt for sharing with us the core event coding algorithm of TABARI, which served as the corner stone for developing EVENTUS ID.

Contents

Chapter 1

Introduction

1.1 Introduction

EVENTUS ID is a system for supervised coding of event data extracted from text written in Spanish. The program is *supervised* as it requires human intervention for developing dictionaries used as search categories and for assessing the accuracy of the coding outcome. This software is capable of *coding event data* based on a system of pattern recognition that provides information on who did what to whom, when and where. The flexibility provided by two event coding algorithms and an event locator protocol fully integrated into the coding process enables EVENTUS ID to identify fine-grained event data at the subnational level. To the best of our knowledge, this is the first program specifically designed for processing short news summaries *written in Spanish*.

EVENTUS ID was initially developed as part of Osorio's doctoral dissertation entitled "Hobbes on Drugs: Understanding Drug Violence in Mexico," which analyzes the dynamics of organized criminal violence in Mexico (Osorio, 2013). The event data approach was necessary for identifying a wide range of violent and non-violent law enforcement actions conducted by government agencies against criminal organizations, as well as a broad menu of violent tactics perpetrated by criminal groups against state authorities and against rival criminal organizations. The project initially intended to use TABARI, a well known event coding protocol created by Schrodt (2009). However, TABARI had been designed for processing text written in English and performed poorly when we tried using it for coding events from text written in Spanish. We approached Schrodt with some inquiries about making TABARI more flexible so it could be better adapted to the particular features of the Spanish language. Schrodt kindly shared TABARI's core coding algorithm with us, and it

became the corner stone for our development of EVENTUS ID in our own way. We are deeply grateful to Philip Schrodt for his generosity and valuable contributions to the progress of conflict research.

EVENTUS ID shares with TABARI some basic characteristics of event coding such as the Deterministic Finite Automata (DFA) approach for string matching and the use of sparse parsing of sentences rather than full syntactical analysis. However, EVENTUS ID and TABARI are quite different in several other respects. As discussed in Chapter 3, EVENTUS ID works with a more flexible set of event coding algorithms that better adapt to the complexities of the Spanish language. In addition, as Chapter 4 shows, EVENTUS ID is capable of identifying the geographic location of events when the information is provided in the source text, a feature not yet integrated in TABARI. Due to the irregularity of verb forms in different tenses in Spanish, we decided not include a stemming application in EVENTUS ID as the one used in TABARI (see Section 3.3). The lack of a stemming function requires the researcher to invest more effort in developing encompassing and detailed dictionaries. In general, the combination of these features and others detailed in this manual contribute to improving the performance of EVENTUS ID for event coding from text written in Spanish.

At the time we were developing the beta version of EVENTUS ID, Leetaru and Schrodt (2013) launched the Global Database on Events, Location and Tone (GDELT), an enormous collection of more than 200 million events generated using TABARI. This unprecedented database immediately sparked a wave of excitement about massive event data using automated coding (Ulfedler, 2013*b*,*a*). After the initial surge of optimism, critics raised concerns about the limitations of GDELT and computer-generated data (Weller and McCubbins, 2014; Hanna, 2014; Steinert-Threlkeld, 2014; Ward et al., 2013). Eventually the debate converged in a more balanced approach recognizing the value of machine-generated data while acknowledging its limitations (Moore, 2014*a*,*b*; Price and Gohdes, 2014).

Among other limitations, critics argue that GDELT has problems of media coverage bias (Beieler, 2013) (for a review of the characteristics and consequences of media bias in conflict research see Davenport (2009) and Davenport and Ball (2002)). However, a crucial aspect of GDELT's media coverage bias that has received scant attention is the fact that it gathers information primarily from news sources written in English.[1] By focusing exclusively on information gathered from English-language news wires, GDELT neglects a vast amount of timely, detailed news published in the

[1]GDELT relies on news reports gathered from the following information sources: AfricaNews, Agence France Presse, Associated Press, Associated Press Online, Associated Press Worldstream, BBC Monitoring, Christian Science Monitor, Facts on File, Foreign Broadcast Information Service, The New York Times, United Press International, and The Washington Post (see section "Data Sources" at `http://gdeltproject.org/about.html#sthash.U65HfSJ3.dpuf`).

native language of a large number of countries. Ignoring non-English written reports reduces the quality of information about the situation on the ground and ultimately degrades the accuracy of the event data outcome. This is particularly problematic if we consider that 91.8 percent of the world population are non–English speakers (Lewis, Simons and Fennig, 2013).

We expect that EVENTUS ID can serve as a useful tool for other researchers to help them develop their own databases of event data from text written in Spanish. In contributing to reducing the language and media coverage bias of machine generated event data, we would have achieved much more than what we originally envisioned.

1.2 Event Data

EVENTUS ID relies on pattern recognition to identify events from text written in Spanish. At its core, an event is a set of information providing a description of someone doing something to someone else. Events are composed of three basic elements:

Source: The actor or perpetrator of the action. Actors are identified by EVENTUS ID as proper nouns in the text.

Action: The specific action carried out by the source. Actions are identified by the system as verb phrases in the text.

Target: The actor towards or upon whom the perpetrator carried out an action.

Besides indicating who (source) did what (action) to whom (target), a more comprehensive description should also indicate when and where the event took place. This provides two additional elements:

Date: The date when the episode occurred.

Location: The specific location of the event.

To detect the source, action and target, EVENTUS ID uses dictionaries of actors and verbs. While reading the text, EVENTUS ID uses the lists of proper nouns and verbs provided by the dictionaries as searching categories to recognize actors and actions. Once these elements are detected, the program puts the textual information into numeric format and stores the codes in a database. While coding events, EVENTUS ID identifies the date of the event using information provided by the file nomenclature. Finally, the program relies on a dictionary of locations to identify the place of occurrence of the episode.

1.3 Overview of Eventus ID Coding Process

The process of event identification process implemented in Eventus ID consists of six stages: information gathering, formatting the text corpus, event coding, event location, validation, and output generation. The first two stages are implemented using the ancillary software Web Text Downloader and Web2Eventus respectively. The process of event identification and event location are carried out in the third and fourth stages. For the sake of clarity, these two stages are discussed separately, but they are integrated into Eventus ID automatically. The fifth stage requires additional human intervention for improving the accuracy of the automated coding protocol in order to achieve the final stage, generating a validated database. The diagram in Figure 1.1 presents all the stages and their main associated tasks. In brief, the process is structured as follows:

Stage 1. Information gathering. The user identifies a set of news resources relevant to the research project. We recommend the use of the program Web Text Downloader and its related procedure for extracting and storing news reports (Osorio and Reyes, 2014*b*). The output of this stage is a collection of news reports in individual files. Since a variety of automated and assisted web scraping techniques are available, Stage 1 is not discussed in this manual.

Stage 2. Corpus of text. The user gathers the collection of news reports to build a text corpus to be input to Eventus ID. The corpus is a large, structured body of text containing the content of all news reports gathered. We recommend the use of the program Web2Eventus and its related procedure for structuring the text corpus according to the format required by Eventus ID (Osorio and Reyes, 2014*a*). The output of this stage is the text corpus to be input to Eventus ID for event coding. This stage is the main topic of Chapter 2.

Stage 3. Event identification using Eventus ID. The user runs Eventus ID to identify event data from the corpus of text. Identifying events requires that dictionaries of actors and verbs have been created. Eventus ID uses the list of actors and verbs as search categories to find the elements of event data (source, action and target) in news reports. To identify events, Eventus ID relies on two event coding algorithms that better accomodate to the characteristics of the Spanish language. The output of this stage is a database containing event data identified in the corpus of text. This stage is discussed in Chapter 3.

Stage 4. Event location using Eventus ID. This stage requires location dictionaries to have been created containing the names of states and municipalities to be used as search categories, along with some filters to enable disambiguation of specific cases. The event location algorithm uses locality names to inspect the corpus of text in order to identify the places where the events occurred. The output of

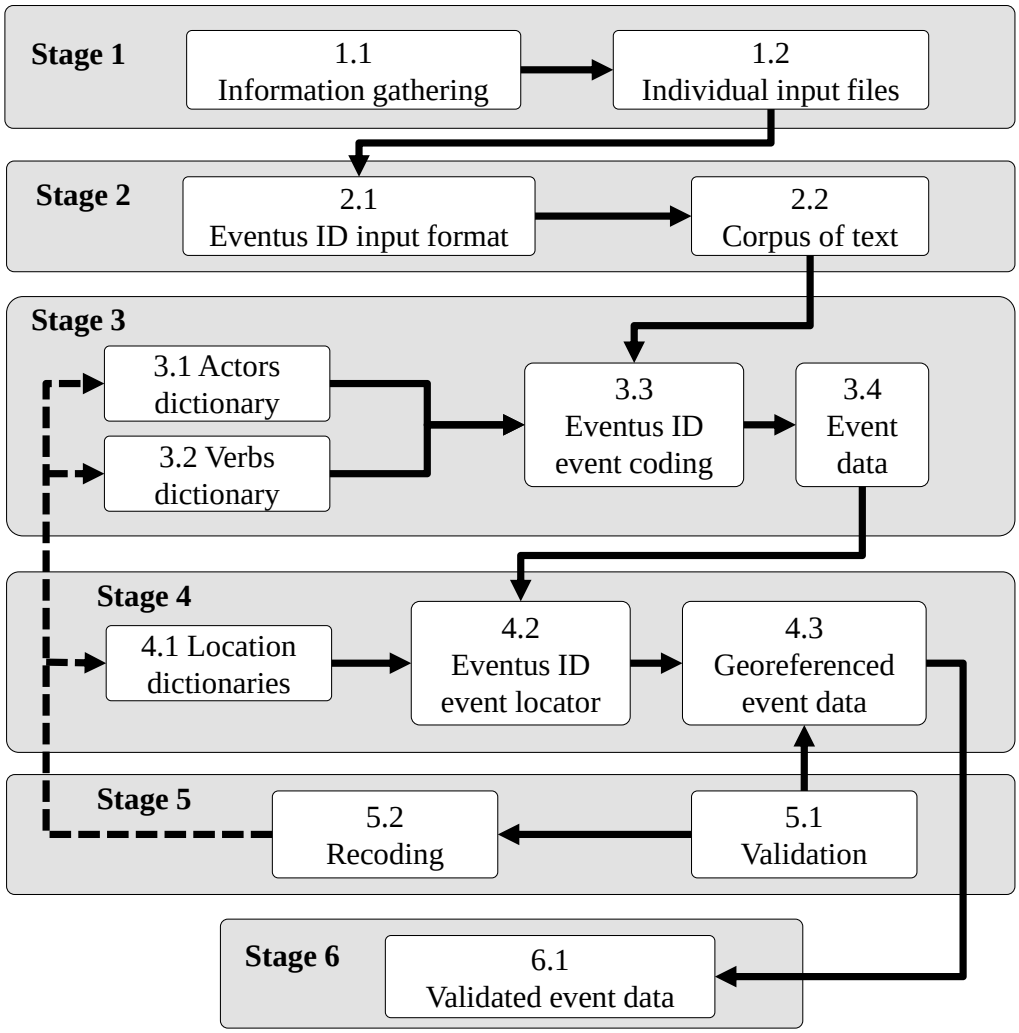

Figure 1.1: EVENTUS ID coding process.

this stage is a database of georeferenced event data. Although analytically distinct, the event identification and the location detection procedures are automatically executed when EVENTUS ID is run. Chapter 4 discusses the specifics of the event location process.

Stage 5. Validation. The user compares a sample of manually coded event data with the events identified by EVENTUS ID. Discrepancies between the manual and computer-generated data provide insights for modifying the actor, verb or location dictionaries. A series of iterations can be expected to improve the accuracy of the automated coding protocol and the validity of the resulting data.

Stage 6. Output. The final output of the coding process is a validated database of georeferenced event data. The validation stage and the output are discussed in Chapter 5.

1.4 Downloading EVENTUS ID and System Requirements

1.4.1 Downloading the Software

Users can access EVENTUS ID version 2.0, the user manual and working files by downloading the demonstration file (Eventus_ID_DEMO.zip) from the following website:

`http://www.javierosorio.net/#!software/cqbi`

1.4.2 System Requirements and Additional Software

1. EVENTUS ID runs on Windows 7, or later versions.
2. EVENTUS ID runs in the command line interface.
    - To launch the command line interface in Windows, click on the Start button in the lower left of the screen, then type the command `cmd` in the search bar and press enter. This will open the command line interface as shown in Figure 1.2.
3. EVENTUS ID requires PERL 5, or any later version to be already installed.
    - We recommend using STRAWBERRY PERL, a canned PERL environment which contains the set of tools and libraries necessary for using EVENTUS ID. STRAWBERRY PERL is available for download at `http://strawberryperl.com/`.
    - Users can find more PERL resources and documentation at `http://www.perl.org/`.
    - To find out which PERL version is installed in your computer, type `perl -v` in the command line. It will display your current PERL version.
4. Users will also need a plain text editor to edit program files. We recommend using NOTEPAD++, a source code editor that supports PERL scripts. NOTEPAD++ can be downloaded at `http://notepad-plus-plus.org/`.

The beta version of EVENTUS ID ran in the Unix environment. However, we decided to adapt Version 2.0 for the Windows operating system. We expect that this will reduce the technological barrier and increase the number of users in Latin America, a region where Unix and Mac operating systems are not as popular as Windows.

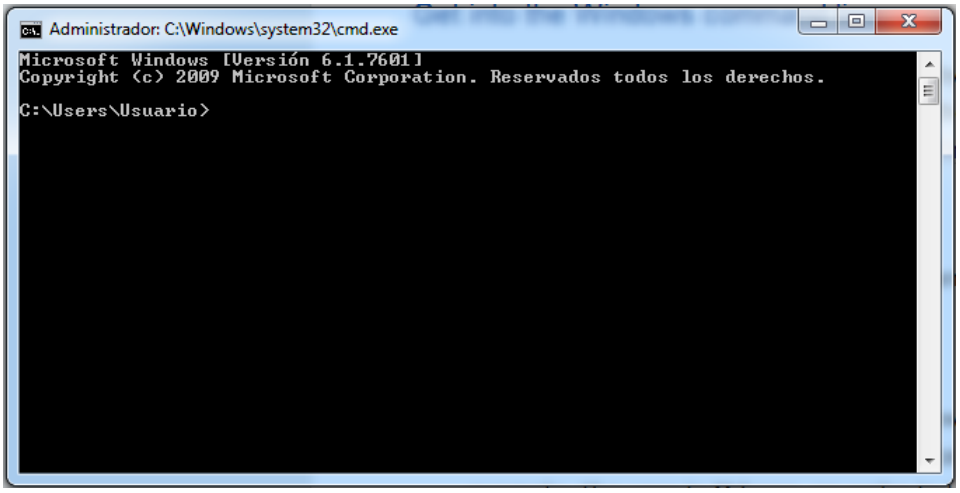

Figure 1.2: Command line interface.

1.5 Font conventions

This manual uses the following typeface convention:

Computer Programs are in Small Caps font.

File names are in Sans Serif font.

Filename extension is a suffix usually two to four characters long indicating the encoding (format) of a file. Extensions begin with a period and are in *italics*. The file formats used in this manual are:

.pl Perl format file

.txt Unicode plain text file

.html Hyper Text Markup Language file

.pdf Portable Document Format file

.csv ASCII text as comma separated values file

`Commands, key entries` and `directories` in the command line interface are in `typewriter` font.

`URL` stands for Uniform Resource Locator and indicates network resources such as websites or email addresses. `URLs` are in `typewriter` font.

`Text content` in input files such as the corpus of text, dictionaries (e.g. actors, verbs, locations) and output files are also in `typewriter` font.

Special symbols:

◇ is the diamond symbol representing a white space in the text.

⟶ is the right arrow representing a space generated by the tabulator (tab) key in the text. The tab symbol is usually not visible in text editors, unless the "Show All Characters" option is activated. NOTEPAD++ users can activate this feature by clicking on the icon ¶.

| is the vertical bar symbol and is used as a marker in the output files generated by EVENTUS ID.

1.6 Share Your Papers and Data

We are committed to providing public goods for innovative research. We want to acknowledge Phillip Schordt, who generously shared TABARI's core algorithm with us, thus providing the cornerstone for developing our won coding protocol. We are truly standing on the shoulders of a giant. In reciprocity, we want to make EVENTUS ID v.2.0 publicly available as free software.

We hope you find EVENTUS ID useful in your research. If you use EVENTUS ID, we would love to post your papers and data at our website. We invite you to share them with us. Ideally, we would like this to promote the development of a community of researchers coding a wide variety of event types from text written in Spanish. There is so much data to be coded in Latin American countries. By making EVENTUS ID software free, we hope others will be able to address important topics by generating valuable new data.

1.7 About the name EVENTUS ID

The name EVENTUS ID comes from joining the Latin word *ēventūs*, which is the plural noun for events, and ID, the abbreviation for identification. In this way, the program name provides a simple and general idea of what the software does: it identifies events. Although EVENTUS ID was originally developed for coding events from text written in Spanish, the program is potentially capable of processing text in other languages that use the Latin script as a writing system. However, we still have to conduct some tests to assess the coding accuracy of the program in other languages.

Chapter 2

The Text Corpus

This chapter describes the format of the text used as the *corpus* in EVENTUS ID. To do so, it explains how to use the ancillary program WEB2EVENTUS for processing previously downloaded news reports and inserting the content into a single document (Osorio and Reyes, 2014*a*). WEB2EVENTUS is available at `http://www.javierosorio.net/#!software/cqbi`.

In brief, WEB2EVENTUS processes each news report by extracting the content, breaking it down by paragraphs, attaching a paragraph counter, formatting the information and storing all the text into a file readable by EVENTUS ID.

Prior to using this program, users must have a set of individual files containing news reports. It is suggested that the users employ the WEB TEXT DOWNLOADER software and follow the procedure described in its manual for extracting news reports from the web.

2.1 Required Software and Files

To process news reports to convert them into EVENTUS ID format, it is necessary to have the following files in the same directory:

1. The program file web2eventus.pl. This program runs on the Windows command line interface and requires PERL 5 or later version.
2. A folder containing the individual files of news reports previously extracted from the web. The example presented in this manual uses news reports stored in Hyper Text Markup Language (*.html*) format, but the software is also capable of processing plain text files (*.txt*).

2.2 News Report File Naming Convention

In order for WEB2EVENTUS to process news report files, individual file names must follow a specific set or naming rules. Complying with this nomenclature is crucial for properly managing files, formatting the corpus of text used in EVENTUS ID and extracting information about the date when an event took place. The nomenclature consists of four main elements:

- Date
- Counter
- Source
- Extension

File names constitute the primary document identification code and should be structured as: yyyymmddccc_SRC.ext, where:

- yyyy is a four-digit number representing the year (e.g. 2009).
- mm is a two digit number representing the month (e.g. 02 for February).
- dd is a two digit number representing the day (e.g. 17).
- ccc is a three-digit number counting the number of reports issued by the same source in a given day. The three digits allow the counter to range from 001 up to 999.
- SRC is a short acronym indicating the name of the information source.
- ext is the extension indicating the format of the file. WEB2EVENTUS can process files in *.html* or *.txt* format.

There should be no duplicate file names. The counter in the nomenclature (ccc) is an effective way of assigning unique file names even if there are a large number of news reports. For convenience, the counter considered in this example is set to three digits, but users can assign as many digits to the counter as they consider necessary.

For example, consider a set of three on-line press releases issued by the Mexican Army (Secretaría de la Defensa Nacional, SEDENA). The first two news reports were issued on August 23, 2009 and the third report on October 17, 2010. According to the nomenclature, the files should have the following names:

```
20090823001_SEDENA.html
20090823002_SEDENA.html
20101017001_SEDENA.html
```

2.3 Using Web2Eventus to Format the Corpus of Text

To create a corpus of text for Eventus ID using previously downloaded news reports, users must follow the following steps:

Step 1: In the command line, type: `perl web2eventus.pl` and then press the Enter key. The command `perl` calls the Perl environment, and the command `web2eventus.pl` launches the program Web2Eventus.

Step 2: Then, the program will then ask the user to enter the path to the folder containing the files to be processed and then press Enter. The directory path can be entered in either of two ways:

- To indicate the absolute path, type: `C:/directory/web/*.html`
- To indicate the relative path, type: `web/*.html`

These command lines both instruct Web2Eventus to process all *.html* files from folder `web`. The absolute path indicates the specific location of the directory in the computer station used for this task. The relative path indicates the software to process the files contained in a sub-folder already hosted in the working directory being used.

Web2Eventus can also process news report files in plain text format. To do so, users just need to replace the extension *.html* with *.txt* in the command line.

If users want to process only one particular file, they can do so by indicating the specific file name as `web/file_name.html`.

Step 3: Finally, the program asks the user to enter the name of the output file (e.g. corpus_DEMO.txt) containing the corpus of text to be used by Eventus ID, and then press enter. Do not forget to include the *.txt* extension in the file name.

2.4 Output File: Corpus of Text for Eventus ID

Web2Eventus generates an output file (corpus.txt) ready to be used in Eventus ID as text corpus. Each line in the output contains the information of one individual paragraph from one news report. The corpus of text presents the information according to the following structure:

`date FileName_P1_P2 | Text`, where:

- `date`: indicates the date of the news report, obtained by extracting this information from the file name entered by the user.
- `FileName`: indicates the file name of each news report.
- `P1`: WEB2EEVENTUS breaks each news report into paragraphs. `P1` is a progressive number indicating the local paragraph counter for each news report. This is useful for identifying a specific paragraph within each news report.
- `P2`: is the global paragraph counter for all news reports comprised in the corpus of text. This counter is useful for quickly locating a specific paragraph in the corpus.
- `|`: the vertical bar symbol (`|`) is a marker indicating the beginning of the text extracted from the paragraph.
- `Text`: the content of each paragraph from each news report is stored in a single line. The length of each line depends on the number of characters in the paragraph.

The corpus of text generated for use by EVENTUS ID consists of a plain text file (*.txt*) whose content is structured in the following way:

```
20130808 20130808001_SRC1_P0_P1 | Lorem ipsum dolor sit amet...
20130808 20130808001_SRC1_P1_P2 | Praesent at sem ac enim ...
20130808 20130808001_SRC1_P2_P3 | Donec sed mattis orci...
20130808 20130808001_SRC2_P0_P4 | Donec velit justo, varius...
20130808 20130808001_SRC2_P1_P5 | Praesent quis felis...
20130808 20130808001_SRC2_P2_P6 | Nunc blandit vitae purus...
20130808 20130808001_SRC2_P3_P7 | Quisque quis lorem sed nunc...
20130921 20130921001_SRC1_P0_P8 | Sed ornare, nisi vitae...
20130921 20130921001_SRC1_P1_P9 | Nulla vel condimentum...
20130921 20130921002_SRC1_P0_P10 | Phasellus porta ipsum eu...
20130921 20130921002_SRC1_P1_P11 | Etiam porttitor vitae odio...
20130921 20130921002_SRC1_P3_P12 | Donec cursus metus vel...
```

In this example, the first three lines represent the paragraphs extracted from a news report issued on October 8, 2013 by source `SRC1`. Lines four to seven show the content from a report issued the same day by a different source, `SRC2`. Notice how the local paragraph counter indicates the number of paragraphs in each news report while the global paragraph counter indicates the total number of paragraphs in the corpus of text. Lines eight to twelve contain the information from two news reports issued by the same source (`SRC1`) on the same day (September 21, 2013). Notice that

the file nomenclature system assigns unique names to each file (`20130921001` and `20130921002`, respectively), which also helps provide unique paragraph identifiers when the local and global paragraph counters are added.

Besides reformatting the text by paragraph, WEB2EEVENTUS identifies phonetic diacritic and emphatic marks on vowels, also known as acute accents (e.g. á, é, í, ó and ú) and substitutes them by their corresponding vowels without accents. Although accent marks constitute an important element of correct spelling in Spanish, several factors motivate the decision to eliminate accents from the corpus. First, it facilitates the task of dictionary development. Since there are a variety of ways of writing special characters in *.html* language, users who want to use text with accents would have to tackle the daunting task of including each word with its different accent codes (e.g. `arrestó`, `arrest&oacute`, `arrest&#243` or `arrest'f3`) in the dictionaries. Second, spelling errors are unfortunately quite common in news reports written in Spanish, as journalists often forget to write the accents in words. Even if the researcher includes words with accents in the dictionaries, low quality text that fails to comply with accent spelling rules would generate coding error. Since EVENTUS ID "reads" text in Spanish without accent marks, this manual does not include accent marks in text examples written in Spanish and denoted in `typewriter` font.

The reformatting process also cleans up the text by eliminating some punctuation signs (e.g. " " : ; ? ! - _). It is important to notice that WEB2EEVENTUS also creates blank spaces to the sides dot "." and comma "," punctuation signs, thus reformatting them as "◇.◇" and "◇,◇", there the symbol ◇ represents a blank space. This characteristic of the corpus of text is crucial for EVENTUS ID to identify the event location as described in Chapter 4 .

2.5 Corpus Example in the Demonstration File

Recent efforts in computerized coding of event data aim to provide reliable and open access of event datasets based on transparent and documented coding protocols and source texts (Open Event Data Alliance, 2014). However, as indicated by Schrodt (2014), license conditions or other intellectual property rights often prohibit researchers from sharing copyrighted source texts, thus hindering the transparency, replicability and validation of the coding protocol.

To address this issue, the text corpus in the EVENTUS ID demonstration file contains *real* press releases issued by the Mexican Army (Secretaría de la Defensa Nacional, SEDENA). These documents were officially issued by Mexican authorities and account for facts and events that took place in the indicated dates and locations. The folder SEDENA_permission in the Eventus_ID_DEMO.zip file contains the official letter authorizing the use of these press releases. In compliance with the requirements of SEDENA, the folder includes a file containing the citations of the documents used in the corpus of text.

Chapter 3

Event Coding

This chapter is divided into five sections. The first section gives the basic steps for running EVENTUS ID. The second and third part describe the characteristics of actor and verb dictionaries in more detail. The fourth section describes the coding algorithms used by EVENTUS ID for event coding. The last section discusses some of the limitations of event coding using EVENTUS ID.

3.1 Event Coding Using EVENTUS ID

This section lists the basic steps for event coding using EVENTUS ID. The specific file names mentioned in this manual are those contained in the Eventus_ID_DEMO.zip file. The content, characteristics and functions of each of these files are further discussed in the different sections of this chapter.

EVENTUS ID requires the following programs and files to be contained within the same directory:

- EVENTUS_ID_2.0.pl is the program file of EVENTUS ID in Perl.
- actors_DEMO.txt is the dictionary listing the actors to be identified as the source or target of an event.
- verbs_DEMO.txt is the dictionary indicating the actions carried out by the actors.
- corpus_DEMO.txt is the corpus of text in the format readable by EVENTUS ID. The corpus must be processed using the program WEB2EVENTUS.

- codigos_Events_DEMO.txt is the output file of numerical event codes generated by the event coding procedure. Alternatively, the user can substitute the numerical coding by the textual codification file (textos_Events_DEMO.txt).
- states_DEMO.txt is the dictionary providing the list of states.
- municipalities_DEMO.txt contains the list of municipalities and towns.
- filters_DEMO.txt is a set of items for disambiguating locations.

With the exception of the EVENTUS_ID_2.0.pl program file, all other files are in plain text format (*.txt*).

There are two ways of running EVENTUS ID. Users can enter the instructions step by step procedure or implement the "quick and dirty" approach. These procedures are outlined in the following sections.

3.1.1 Running EVENTUS ID Step By Step

Users can run EVENTUS ID by implementing the following steps:

Step 1. Open terminal: Launch the command line interface in Windows and move to the directory containing the files indicated above.

Step 2. Launch EVENTUS ID: In the command line, type: `perl EVENTUS_ID_2.0.pl` and then press the Enter key. This will start running EVENTUS ID using the perl environment.

Step 3. Enter the list of sources: Enter the file name of the actors dictionary used to identify the source actor. Type: `actors_DEMO.txt` and press Enter.

Step 4. Enter the list of targets: Enter the actor dictionary used to identify the targets of actions. The DEMO files uses the same list of actors to identify both the source and target of an event. In this case, type `actors_DEMO.txt` again and press Enter. If necessary, EVENTUS ID has the flexibility to allow users to enter a different list of actors to identify the targets of an event. In that case, type the name of a specific target actor dictionary.

Step 5. Enter the list of actions: Enter the verb dictionary to identify the actions. Type: `verbs_DEMO.txt` and press Enter.

Step 6. Enter the corpus: Enter the corpus of text used for event coding. Type: `corpus_DEMO.txt` and press Enter. The text corpus must be previously created

using the program WEB2EVENTUS and formatted according to the specifications outlined in Chapter 2.

Step 7. Enter the list of municipalities: Enter the dictionary of municipalities used for identifying event locations. Type: `municipalities_DEMO.txt` and press Enter.

Step 8. Enter the list of states: Enter the states dictionary used for identifying event locations. Type: `states_DEMO.txt` and press Enter.

Step 9. Enter the list of filters: Enter the filter dictionary for disambiguating event locations. Type: `filters_DEMO.txt` and press Enter.

Step 10. Enter the name of the event coding output: Type: `Events_DEMO.txt`, then press the Enter key. EVENTUS ID uses that name to automatically generate two files, one containing the numeric codes of the elements identified in the corpus and another containing the actual words identified in the text. Based on the name provided, the program generates the file codigos_Events_DEMO.txt including the numeric codes and the file textos_Events_DEMO.txt containing the text information. The former is primarily intended to be used for conducting statistical analysis, and the latter is intended to assist users in validating the accuracy of coding protocol.

Step 11. Select the algorithms used for event coding: The user has the option of selecting distinct coding algorithms to identify event data in the text corpus. As discussed in detail in section 3.4, EVENTUS ID includes a general sequence algorithm and a partial sequence algorithm for extracting event data from different syntactical structures. Users can enter either 1 or 2 to select the desired coding scheme. Type option `1` and press Enter for coding events using only the general sequence algorithm. Type option `2` and press Enter to use both the general and the partial sequence algorithms for event coding.

The final output generated by EVENTUS ID contains information about the source, action, target, date and location of each event, thus providing an encompassing description of who did what to whom, when and where.

3.1.2 Running EVENTUS ID "Quick and Dirty"

Users can run EVENTUS ID in just a couple steps without manually entering all the individual files specified in the step by step procedure. Two files are necessary to implement the "quick and dirty" approach:

- EVENTUS_ID_2.0.pl is the program file of EVENTUS ID in Perl.

- config_DEMO.txt is the configuration file containing the name of all files necessary for automatically running EVENTUS ID.

The file names contained in the config_DEMO.txt file correspond to the steps mentioned in the previous section and must be listed in the following order:

```
actors_DEMO.txt
actors_DEMO.txt
verbs_DEMO.txt
corpus_DEMO.txt
municipalities_DEMO.txt
states_DEMO.txt
filters_DEMO.txt
Events_DEMO.txt
2
```

To run EVENTUS ID by the "quick and dirty" method, users have to carry out these two steps:

Step 1. Open terminal: Launch the command line interface in Windows and move to the directory containing the files listed above.

Step 2. Launch EVENTUS ID and the configuration file: In the command line, type: `perl EVENTUS_ID_2.0.pl config_DEMO.txt` and then press the Enter key.

The last entry in the config_DEMO.txt file corresponds to option 2 in Step 11, which relies on both the general and partial coding algorithms to identify events in the text corpus.

The rest of this chapter discusses the characteristics of the actor and verb dictionaries and the different event coding algorithms used in EVENTUS ID. The procedure for identifying the geographic location of the events is presented in Chapter 4.

3.2 Actor Dictionary

EVENTUS ID uses the actor dictionary to identify both the source and the target of an event. As mentioned before, the program has the flexibility to detect these two elements using different dictionaries. However, for the sake of simplicity, this section discusses the characteristics of the actor dictionary assuming it will be used for coding both the source and the target. The actor dictionary consists of a list of proper nouns which can refer to perpetrators and victims of various types of violence. Proper nouns serve as searching categories enabling actors in the text corpus to be identified. Each item in the actor dictionary is associated with a numeric code that corresponds to the actor's main group and subgroup as determined by the researcher coding protocol.

The actor dictionary file is in plain text format (*.txt*). Actors' names composed of multiple words are separated by an underscore "_" to help EVENTUS ID searcg for the words in the text. The underscore "_" should also be the last element of each word in the actor dictionary. The program does not require words in the text corpus to be separated by an underscore. EVENTUS ID "reads" the corpus containing words separated simply by blank spaces as in any regular text, but it uses the concatenated words of the dictionaries to identify patterns in the corpus. Once the name of an actor matches the content of the text, its corresponding code is stored. The following lists present an example of the actor dictionary in both English and Spanish:

English:

```
army_troops_                          [202051]
police_officer_                       [204021]
member_of_a_criminal_organization_    [601060]
cocaine_                              [801022]
AK_47_                                [901013]
```

Spanish:

```
tropas_del_ejercito_                    [202051]
oficial_de_policia_                     [204021]
miembros_de_una_organizacion_criminal_  [601060]
cocaina_                                [801022]
AK_47_                                  [901013]
```

3.3 Verb Dictionary

The verb dictionary consists of a list of verb phrases referring to actions conducted by a source or directed against a target. Like the elements of actor dictionary, the list of verbs must be contained in a plain text file (*.txt*). Each item in the verbs dictionary is followed by a numeric code for the corresponding action category assigned by the user. In some occasions, actions can be adequately captured with a single verb phrase. However, the complexities of natural language usually require using more complex sentences for accurately identifying the actions of interest. As in the actor dictionary, actions described by more than one word are separated by an underscore "_", which is also required at the end of each verb item.

The following list is an example of the verbs dictionary in English and Spanish:

English:

```
attack_                        [88101]
attacked_                      [88101]
- were_*                       [99101]
arrest_                        [88104]
- under_*                      [88104]
combat_                        [88101]
- strengthen_the_*against_     [- - -]
```

Spanish:

```
atacar_                      [88101]
atacados_                    [88101]
- fueron_*                   [99101]
arresto_                     [88104]
- bajo_*                     [88104]
combate_                     [88101]
- fortalecer_el_*contra_     [- - -]
```

In order to improve the accuracy of the coding protocol, users might consider using a variety of verb tenses to refer to the same action. As shown in the example above, the verbs "`attack`" and "`attacked`" have the same code `88101`. Some verbs are followed by a set of associated words that refine the meaning of the verb in its context. In these cases, the "*" symbol serves as a wild card indicating where the preceding verb itself should appear in the phrase. In the example above, EVENTUS ID automatically inserts the verb "`attacked`" into the item `were_*` and looks for the verb phrase "`were attacked`" in the corpus of text. The code for "`attack`" is slightly different than the code for "`were attacked`." Both cases have the root code `101` but they differ in the prefix; the former starts with `88` and the latter with

99. The prefix 99 is useful as a disambiguation system to indicate a verb in passive voice. This reference serves as a hint for the validation and recoding processes, as well as for statistical analysis of the data. We found this coding system useful for processing news reports in Spanish because, in contrast to writing recommendations in English, the use of passive voice is highly common in journalistic reports written in Spanish (Rodríguez, 2001).

As mentioned earlier, EVENTUS ID uses a pattern recognition coding scheme similar to that implemented by TABARI. However, a key difference between these two protocols lies in the way they handle verb phrases. TABARI is designed for coding in English, whose grammatical rules allow for simple general ways of combining the subject, verb and object in a sentence. For example, the forms of most regular verbs in English vary only by adding "s," "ed," or "ing" to the end of the infinitive stem. Another characteristic of English is that the gender and number of the noun do not affect the verb form (excepting only the third person singular present). This means that "you," "he," "she," "we" or "they" can be mostly combined with the different verb forms almost indistinctly. Based on these simple and general grammatical rules, TABARI incorporates a *stemming* algorithm to automatically identify all the different verb tenses from a stem. Using the verb "to arrest" as an example, Table 3.1 shows that English allows the stem "arrest" to be easily conjugated for different verb tenses, gender and numbers by simply adding a suffix at the end of the stem. TABARI's takes advantage of this grammatical characteristic to readily identify a variety of verb forms.

Although TABARI's stemming facility is very convenient for coding in English, this feature can generate a substantial amount of error when coding in Spanish. Using an English-based stemming algorithm is not appropriate for event coding in Spanish because verb tenses in the latter do not end with "s," "ed" or "ing." Verb conjugation in Spanish is much more varied, and using an English-based stemming algorithm would require more than simply "tweaking" the algorithm. Table 3.1 shows that the ending part of the verb "*arrestar*" (to arrest) is very different across the various combinations of verb tenses, number and gender. Given the complexity of conjugation in Spanish, EVENTUS ID does not include a stemming algorithm. The "shortcuts" that might be useful for coding in English would be counterproductive in Spanish. Unfortunately, reducing the propensity to generate error through a stemming process comes at a cost. In order to improve the accuracy of the coding protocol, EVENTUS ID requires the user do develop large and detailed verb dictionaries considering a variety of verb tenses for different gender and number of subjects.

Table 3.1: Verb tenses in English and Spanish

Person	Indicative		Subjuntive		Gerund	Past passive voice
	Present	Past	Present	Imperfect		
English						
I	arrest	arrested	arrest	arrested	arresting	was arrested
you	arrest	arrested	arrest	arrested	arresting	were arrested
he, she	arrests	arrested	arrests	arrested	arresting	were arrested
we	arrest	arrested	arrest	arrested	arresting	were arrested
you	arrest	arrested	arrest	arrested	arresting	were arrested
they	arrest	arrested	arrest	arrested	arresting	were arrested
Spanish						
yo	arresto	arresté	arreste	arrestara o arrestase	arrestando	fui arrestado
tú	arrestas	arrestaste	arrestes	arrestaras o arrestases	arrestando	fuiste arrestado
ella, él, usted	arresta	arrestó	arreste	arrestara o arrestase	arrestando	fue arrestada o fue arrestado
nosotros	arrestamos	arrestamos	arrestemos	arrestáramos o arrestásemos	arrestando	fuimos arrestados
vosotros	arrestáis	arrestasteis	arrestéis	arrestarais o arrestaseis	arrestando	fuisteis arrestados
ellas, ellos, Uds.	arrestan	arrestaron	arresten	arrestaran o arrestasen	arrestando	fueron arrestadas o fueron arrestados
vos	arrestás	arrestaste	arrestés	arrestaras o arrestases	arrestando	fuiste arrestado

Developing actor and verb dictionaries requires a reiterated process of learning, refinement, knowledge accumulation, detailed reading and, feedback from the validation process. This repeated process of coding, verification and recoding allows the dictionaries to be fine-tuned by adding actors and verbs or modifying existing ones. However, it is important to note that it is not possible to develop "perfect" dictionaries including "absolutely all" possible actors or verb phrases and capable of achieving a 100 percent coding accuracy. As noted by Grimmer and Stewart (2013) all quantitative models of language are flawed, but some are useful.

In some instances, words referring to verbs or actors of interest can have ambiguous meanings depending on specific sentence constructions that the user wants to avoid. For those cases, EVENTUS ID offers the possibility of ignoring some pieces of information that could generate coding error. Following the example of the verbs dictionary presented above, users interested in coding violent confrontations between different actors might include in the dictionary the verb phrase "`combat`", to which they assign the code number `[88101]`. However, news reports might include a sentence stating that "`Government efforts aim to strengthen the combat against criminal organizations`," a common closing line in government press releases. In this context, the word "`combat`" has a metaphoric meaning that does not refer to a specific event of armed combat between government forces and members of criminal organizations. In order to avoid erroneously coding words of interest contained in metaphoric sentences, EVENTUS ID enables a mechanism for disambiguation. In those cases, users might consider assigning the code `[- - -]` to those sentences that they want to be ignored. This null code is applicable for items in both the actor and verb dictionaries and gives EVENTUS ID the instruction to skip that specific sequence of words.

3.4 Event Coding Algorithms

In order to code events from the text corpus, EVENTUS ID uses two pattern recognition algorithms: the *general sequence algorithm* codes events that comply with the full source–action–target structure, and the *partial sequence algorithm* codes incomplete events in the verb–target structure. Both algorithms use the principles of the *sparse parsing* technique originally developed by Schrodt in KEDS and later refined in TABARI. The sparse parsing method uses the actor and verb dictionaries as searching criteria to identify only the relevant parts of the text that correspond to an event, while the rest of the text is ignored for coding purposes.

Based on the nomenclature and corpus format discussed in Chapter 2, both coding algorithms first identify the date of the event (`date`) and the document and paragraph identification label (`FileName_P1_P2`) from the beginning of each line in the

text corpus. Each algorithm then uses its own scheme to recognize the elements of the event contained in the corpus. Table 3.2 describes the steps undertaken by each algorithm for event coding. EVENTUS ID then saves the outcome of the coding process in a plain text file. Each line in the outcome file contains the set of components corresponding to the coded event, separating its elements by tabs and ordering them. As a result, the general and partial sequence algorithms respectively generate the following outcomes:

`date` → `FileName_P1_P2` → `actor1` → `verb` → `actor2`

`date` → `FileName_P1_P2` → → `verb` → `actor2`

Table 3.2: EVENTUS ID coding algorithms

General sequence algorithm	Partial sequence algorithm
1) Search for the actor	1) Search for the verb
- Load the actor dictionary - Start reading the text - Search for the longest actor first - When an actor is found, store as `actor1` - Pause the search where the actor is found	- Load the verb dictionary - Start reading the text - Search for the longest verb first - When a verb is found, store as `verb` - Pause the search where the verb is found
2) Search for the verb	2) Search for the actor
- Load the verb dictionary - Resume reading from previous pause - Keep reading the text - Search for the longest verb first - When a verb is found, store as `verb` - Pause the search where the verb is found - If no verb is found, go to Step 4	- Load the actor dictionary - Resume reading from previous pause - Keep reading the text - Search for the longest actor first - When an actor is found, store as `actor2` - Pause the search where the actor is found - If no actor is found, go to Step 3
3) Search for the actor	3) Save the event
- Reload the actor dictionary - Resume reading from previous pause - Keep reading the text - Search for the longest actor first - When an actor is found, store as `actor2` - Pause the search where the actor is found	- Save `[- - -] [verb] [actor2]` in database - If no actor is found, save the event as `[- - -] [verb] [- - -]` in the database - Start again from Step 1
4) Save the event	
- Save `[actor1] [verb] [actor2]` in database - If no verb is found, save the event as `[actor1] [- - -] [- - -]` in the database - Start again from Step 1	

3.4.1 General Sequence Algorithm

The general sequence algorithm identifies events that follow the source–action–target structure. In order to code an event with this algorithm, all three elements must appear in a sentence in the required order. Consider the following sentence in both English and Spanish (as mentioned in Section 2.4, Spanish text examples in this manual deliberately omit accents):

English:

`Army troops arrested a member of a criminal group.`

Spanish:

`Tropas del ejercito arrestaron a miembro de una organizacion criminal.`

In this example, all the three elements of the event are present in the sentence in the required order. In consequence, the general sequence algorithm identifies "`Army troops`" as the source, ""`arrested`" as the action and "`member of a criminal group`" as the target. EVENTUS ID then codes the event in numeric format in the database as:

`202051` → `88104` → `601060`

Since sparse parsing only focuses on the relevant parts of the text based on the words provided by the actor and verb dictionaries, the text could be more verbose without affecting the result of the coding. Consider the following paragraph:

English:

`In a press release issued yesterday, the Mexican government announced that troops deployed in the municipality of San Luis Rio Colorado, Son. seized 227 packages of marijuana with a total weight of two tonnes and 250 kilograms while patrolling rural roads in the area.`

Spanish:

`En un comunicado de prensa emitido el dia de ayer, el gobierno mexicano informo que tropas destacamentadas en el municipio de San Luis Rio Colorado, Son. decomisaron paquetes de mariguana con un peso total de dos toneladas y 250 gramos, mientras patrullaban caminos rurales del area.`

Despite the wordiness of the paragraph, sparse parsing allows EVENTUS ID to recognize the key components of the event and code the "`troops`" as the source, "`seized`" as the action and "`packages of marijuana`" as the target.

As mentioned above, journalists in Mexico often use passive voice and present indicative to write their news reports. Passive voice increases the grammatical complexity of a sentence by inverting the order of the subject and object. Consider the following sentence:

English:

`A member of a criminal organization was arrested by Army troops.`

Spanish:

`Un miembro de una organizacion criminal fue arrestado por tropas del ejercito.`

In this example, all the three elements of the event are present in the sentence. However the subject and object appear in reverse order. According to the general sequence algorithm, EVENTUS ID identifies "`member of a criminal group`" as the first actor, the verb phrase "`was arrested`" as the action, and "`Army troops`" as the second actor. The output of this event after coding is:

`601060` → `99104` → `202051`

However, this could be interpreted as "a member of a criminal group arrested elements of the Army," which is not the idea expressed in the text. As discussed in Section 3.3, the verb dictionary adds prefix `[99]` to codes corresponding to verb tenses in passive voice. Later, the recoding process (discussed in Chapter 5) uses this prefix as a cue to correct the coding directionality caused by use of the passive voice. In this way, dictionary development, the coding algorithm and the recoding scheme work together to disentangle more complex grammatical structures and reduce coding error in the database. When the recoding rules have been applied, the event is correctly coded as:

`202051` → `88104` → `601060`

As shown in Table 3.2, the general sequence algorithm begins by searching for the first actor in the sentence. Once it is found, it switches to the verb dictionary and looks for the action. However, sometimes news reports mention a series of items which are not followed by a verb. This particularly common in government press releases including a list of items. For example consider the following paragraph:

English:

```
Army troops arrested a member of a criminal group.  Troops seized 6
kilograms of cocaine and other items:
372 packages of Clindamycin phosphate,
two AK-47s,
one R-15 assault rifle,
ammunition.
```

Spanish:

```
Tropas del ejercito arrestaron a miembro de un grupo criminal. Las tropas
decomizaron 6 kilogramos de cocaina y otros articulos:
372 paquetes de fosfato de clindamycina,
dos AK-47,
un rifle de asalto R-15,
municiones.
```

Applying the general coding algorithm, EVENTUS ID recognizes "`Army troops`," "`arrested`" and "`member of a criminal group`" as the first set of source, action and target. Next, it identifies "`troops`" "`seized`" "`cocaine`" as the second set of source, action and target in the paragraph. In the next line, the algorithm starts searching for an actor and identifies "`Clindamycin phosphate`. Since there are no more verbs or actors in the line, the algorithm jumps to the next line and starts searching again for the first actor. In that way, the coding procedure continues reading the items on the list and detects " "`AK-47`," "`R-15 assault rifle`" and "`ammunition`" as independent items not followed by corresponding actions. The output of this paragraph is:

```
202051 → 88104 → 601060
202051 → 88202 → 801022
604021 → →
901013 → →
901014 → →
901021 → →
```

As this example illustrates, the general coding algorithm starts searching for the first actor and then continues to look for the verb and the second actor. If the later two are not found in the text line, the coding algorithm maintains the first actor already identified and jumps to the next line to resume the search. In consequence, EVENTUS ID is capable of extracting more detailed information from complex events, thus better reflecting the multidimensionality of conflict. Some users might find useful the information extracted from the list of items useful. As discussed in Chapter 5, in those cases users might develop a recoding scheme to fill the information in the incomplete coding lines.

3.4.2 Partial Sequence Algorithm

The partial sequence algorithm of EVENTUS ID is useful for identifying more complex grammatical structures such as sentences using present indicative tenses. In general, the present indicative is used similarly in English and in Spanish. However, the present indicative tense is more frequently used in Spanish. In English, the present progressive tense is a finite form of the verb that has the mood, tense, and person clearly defined. For example, in the sentence "they are arresting a criminal" the verb "to arrest" is conjugated in the indicative mood, present progressive tense, third person plural. The present progressive sentence that literally corresponds to this example in Spanish is "*ellos están arrestando a un criminal*". However, in Spanish one would simply use the present indicative tense; thus the sentence would read "*arrestan a un criminal.*" As shown in Table 3.1, this conjugation of the verb "*arrestar*" (to arrest) corresponds to the third person of the present indicative.

The present indicative is formed by removing the infinitive ending of the verb (e.g. taking out the final "ar" from the verb "arrestar") and replacing it with an ending that indicates the person performing the action. What makes the analysis in Spanish more complex is that the conjugation already gives information about the person as part of the verb, and in consequence the subject of the action is often omitted from the sentence. To make things even more complex, the present indicative is often used for referring to events that occurred in the *past* (historical present). Thus while the sentence "*arrestan a un criminal*" literally refers to an action carried out in the present, it also refers figuratively to a past event.

The use of present indicative is a common grammatical structure in journalistic narratives in Spanish media (Martínez, Miguel and Vázquez, 2004; Alcoba Rueda, 1983; Nadal Palazón, 2009). According to Guízar García (2004), 73 percent of news headlines in Mexico use the present indicative verb form. There are several possible reasons for this frequent use. The first might be editorial. Since the present indicative tense usually omits the subject, sentences using this conjugation tend to be shorter, making the present indicative more efficient in terms of printing space. Since newspapers have strict space limitations – determined by the size of the paper they use for printing and the number of pages – editors may favor the use of present indicative tenses for making news reports shorter, which might allow for more reports to be included in daily editions. The second reason may be related to a stronger psychological impact on readers. Sentences written in present indicative usually start with the verb, which draws the reader's attention to the action that took place. In addition, the present verb form may give a sense of immediacy to the event. In this way, editors and journalists often use sensationalist – sometimes lurid – verbs to craft headlines to hook the readers.

The partial sequence algorithm helps code sentences in which the verb is conjugated in present indicative tense. This feature is particularly useful for event coding from

text written in Spanish because this verb tense is very common in Latin American media. Since the translation of present indicative from Spanish into English obscures the nuances of this verb form, the next example is presented in Spanish. Consider the following sentence:

Spanish:

`Arrestan a un criminal`

In this sentence, the subject of the action is omitted because of the conjugation of the verb in present indicative tense. In the absence of a first actor, EVENTUS ID uses the partial sequence algorithm to identify "`arrestan`" (to arrest) as the action and "`a un criminal`" (a criminal) as the second actor. In consequence, the algorithm generates the following event coding:

$\diamond \rightarrow$ `88104` $\rightarrow$ `601060`

As discussed in Chapter 5, users can develop recoding protocols to fill in the missing part of the event by assigning a default source actor to specific types of actions.

3.5 Limitations of Event Coding

A language is a system—a set of ordered rules—that enables users to structure symbols for reference or representation purposes. Different languages (e.g. English, Spanish, Chinese, mathematical, chemical, gesture, chromatic, etc.) use different sets of rules and symbols to represent their objects of interest. In natural language (phonetic and written), these symbols of representation are words. Words are abstractions that constitute the building blocks of language and constitute the key elements used for reasoning and knowledge. Natural language is often highly complex, and computerized methods of textual analysis fall short in accurately capturing the abstractions represented through language. However, despite this limitation, automated-coding protocols can be valuable for specific research objectives.

Based on the analysis carried out by Schrodt and Gerner (2012) on the advantages and shortcomings of human and automated coding approaches, Table 3.3 compares the trade-offs between manual and supervised methods across four main issues. The first group shows trade-offs relating to the characteristics of the coding project. Supervised textual annotation is better suited for processing large volumes of documents, whereas human coding is more appropriate for small scale projects. Automated coding has the advantage of allowing researchers to recode the same documents in repeated coding periods. Supervised machine coding also allows researchers the possibility of easily modifying or updating dictionaries and recoding

the entire set of documents with the new dictionaries. This also allows for easily updating or expanding the project by processing new information. In contrast, coding projects relying on humans usually carry out the coding stage only once because recoding would require substantial resources in terms of time and labor. Sometimes researchers using manual methods discover limitations or problems in their dictionaries when the coding project is at an intermediate or advanced stage. In such cases, modifying the dictionaries for the rest of the project would jeopardize internal consistency, but not modifying them would mean carrying the dictionary problems or limitations through to the end of the project. Another option would be to modify the dictionaries and restart the coding from the beginning. However, scarcity of resources often makes it impractical to recode or update projects using manual methods.

Table 3.3: Comparative advantages of manual and supervised annotation methods

Criterion	Manual coding	Supervised coding
Coding project		
Volume of documents	Small	Large
Coding period	Once	Repeated or continuous
Recoding possibility	No	Yes
Updating possibility	Difficult	Easy
Dictionary modification	Not recommended	Easy
Content of interest		
Coding unit	Entire document	Sentence or paragraph
Syntax characteristics	Complex	Simple
Content of interest	Metaphoric or idiomatic	Literal
Bias concerns		
Sources of coder bias	Several	Unique
Concern of inter-coder reliability	Considerable	Non-existent
Bias caused by coder fatigue	Considerable	Non-existent
Possibility of information bias	Considerable	Minimum
Feasibility of the coding project		
Coding time	Slow	Fast
Labor and financial demands	High	Low

The second group of trade-offs refers to the specific content of interest. Automated coding is more appropriate when the coding unit consists of sentences or short paragraphs with simple syntax, and when the researcher is interested in the literal content of the text. In contrast, manual methods are more appropriate for analyzing the overall content of entire documents and when the coding process requires analytical abstraction or metaphorical analysis.

The third group refers to concerns of bias introduced by coders or by the information sources. Measurement validity is a central concern in both quantitative

and qualitative research (Adcock and Collier, 2001). Systematic coding error may generate measurement bias, which could lead to erroneous conclusions (Collier and Brady, 2004; King, Keohane and Verba, 1994; Geddes, 2003). Coding based on manual methods usually requires human coders to be trained to understand and apply the rules of the coding protocol. Unfortunately, these efforts are not sufficient to guarantee compliance with the coding rules since human coders often apply subjective assessment and heuristic principles to classify information (Tversky and Kahneman, 1974). Increasing the role of coder interpretation and judgment affects the coding outcome and reduces inter-rater reliability (Sipes, 1976; Harvey, 2008), a fact largely neglected by researchers (Coppedge and Wolfgang, 1990; Rohner and Katz, 1970). As noted by Baumgartner, Jones and MacLeod (1998), coder fatigue is another important source of bias in manual coding. Large projects usually involve teams of coders reading vast volumes of information. Often individual motivation and attention diminish as boredom increases. Fatigued coders tend to simply skim through the materials they are supposed to read carefully, thus potentially missing some important parts of the information, and they tend to be less meticulous in the application of complex coding rules. In contrast, automated coding methods eliminate problems of inter-coder reliability by systematically and consistently applying the same coding criteria. Nevertheless, the dictionary developed by the researcher remains a potential (yet unique) source of coder bias in automated coding. In addition, tiredness and boredom are not a concern in computer-generated databases as the machine never tires.

Another source of bias may come from the information sources used in the coding project. Due to limited resources and time constraints, manual coding projects tend to rely on a reduced set of information sources. In conflict research, newspapers remain the dominant source of information for studying violence. However, different sources of information may cover the same events from very different perspectives, thus generating important consequences for the inferences drawn from the evidence reported in those sources (Davenport and Ball, 2002; Davenport, 2009). Automated coding reduces the effect of specific newspapers by simultaneously processing a large number of information sources. Increasing the number of information sources reduces the risk of under-reporting due to coverage bias and helps minimize ideological bias caused by specific political views.

Finally, the fourth group of trade-offs refers to the resources required by different coding strategies. Manual coding usually requires a substantial investment in terms of time, labor and financial resources. Unfortunately, research projects often face significant constraints that have to be weighed when assessing the overall feasibility of the project. Machine-based protocols offer an alternative that substantially reduces the time and financial demands for some types of coding projects that may increase the feasibility of research endeavors when there are budgetary constraints. However, as reflected in Table 3.3, automated coding may not be the best strategy

for all projects and researchers have to carefully evaluate the trade-offs between manual and machine-assisted methods.

Natural language processing concerns the computer-aided analysis of (natural) language produced by humans. Due to the inherent complexities and irregularities of natural language, there is no single best automated textual analysis method (Grimmer and Stewart, 2013). The relevance and accuracy of a method largely depend on the research objective. In projects focused on recognizing specific word categories in the text corpus, supervised textual annotation requiring human involvement or supervision tends to generate more accurate results than fully automated methods (Bigert, 2005). As discussed by other authors (Schrodt, 2009; Schrodt and Gerner, 1994, 2012), the quality of supervised event coding methods depends on three key factors; the effectiveness of the coding algorithm, the accuracy of the dictionaries, and the quality of the text.

Despite the flexibility of coding procedures and the development of encompassing dictionaries, Eventus ID is not a silver bullet. Therefore, users are advised to adjust their expectations. In particular, Eventus ID has the following specific limitations:

- The flexibility of the general and partial event coding algorithms implemented in Eventus ID constitute an improvement over more rigid coding protocols. This feature is particularly useful for improving the accuracy of event coding from text written in Spanish. However, as discussed before, the lack of stemming applications in Eventus ID demands a greater effort from the user in developing detailed actors and verbs dictionaries. Having precise, comprehensive lists of actors and verbs is therefore crucial for effective coding.

- The complexity of grammatical rules and the intricacy of semantic constructions of text written in Spanish represent an important challenge for the accuracy of the coding protocol. Unfortunately, the quality of text is usually beyond the control of the researcher. In consequence, there are limits to the accuracy of event coding text in Spanish using Eventus ID. Therefore, it is crucial for researchers to validate the precision of their automated text analysis, a topic discussed in Chapter 5.

- It is important to remind users that Eventus ID is designed to identify discrete events contained in the text source. In consequence, the performance for detecting specific counts mentioned in the text is limited, especially if those specific quantities are not specified in the dictionaries. For example, consider the following sentence "`The Army seized 227 grams of cocaine`." Given a basic set of dictionaries, the program would be able to identify "`The Army`" as the source, "`seized`" as the action and "`cocaine`" as the target.

However, if the actors dictionary lacks an explicit entry specifying "`227 grams of cocaine`", the program would not be capable of identifying this precise quantity. Users interested in detecting specific quantities mentioned in the text should develop their own detailed dictionaries including explicit counts for each of the items under study. As expected from this example, developing a list of quantities from 0 to a possible maximum number, expressed in different units (e.g. grams, kilograms, pounds, tonnes, pills, packages, etc.), for a wide variety of drugs (e.g. cocaine, opium, LSD, etc.) can easily turn into an daunting task.

Chapter 4

Event Location

The features of Eventus ID described in Chapter 3 enable pieces of information to be extracted from the source text that describe who did what to whom and when the event took place. However, in order to provide a complete account of the event, it is also crucial to know where the episode occurred. Eventus ID is especially suited for addressing this need as the program is capable of identifying the location of event data at two sub-national levels of analysis (state and municipal) when the information is provided in the text corpus. Having georeferenced event data allows fine-grained analysis of the interactions between actors across time and space. Although the event location function is discussed separately in this chapter, this facility is fully integrated into Eventus ID and is performed automatically when the program is run as indicated in Section 3.1.

In general terms, the event location protocol works as follows. Eventus ID uses the output file generated in the event identification procedure and searches for the location of the event in the groups of text. The software uses dictionaries of states and municipalities in order to identify the locality mentioned in the original source. In addition, due to the complexities of identifying localities, the event location protocol includes a filter dictionary for disambiguation that prevents certain words or phrases from being erroneously classified as geographic references.

4.1 Location Dictionaries

Georeferencing event data using Eventus ID requires two different location dictionaries, one with a list of states and the other with the list of municipalities. The program uses these dictionaries as categories for pattern recognition of event locations in the text corpus. Both location dictionary files must be in plain text format

(*.txt*). Each line must begin with the numeric code, followed by textual name of the corresponding location. These two elements should be separated by tabs (→). The lists of states and municipalities and their corresponding codes used in this example and in the DEMO file come from the National Geostatistical Framework generated by the Mexican census bureau, *Instituto Nacional de Estadística y Geografía* (INEGI) (2011). Using official locality codes enables the event data generated to be readily cross-referenced with other databases using the same codes for describing demographic, economic or geographic attributes of each location. Of course, users can use their own location dictionaries and codes.

The following list shows an example of the states dictionary:

```
1 → Aguascalientes
2 → Baja California
3 → Baja California Sur
4 → Campeche
5 → Coahuila
```

This is an example of the list of municipalities:

```
1002 → Asientos
2004 → Tijuana
3001 → Comondu
4010 → Calakmul
5025 → Piedras Negras
```

The need for disambiguation implies the use of the filter dictionary in order to prevent the event location algorithm from identifying false positives. In the original research on drug violence that motivated the development of EVENTUS ID, (Osorio, 2013) realized that location ambiguity problems can emerge from the fact that some criminal organizations are named after the states or cities where they operate. That is the case of "El Cartel de Sinaloa," "El Cartel de Tijuana," "El Cartel de Juarez" and "El Cartel de Jalisco Nueva Generacion," among others. EVENTUS ID uses the filter dictionary to reduce the risk of the algorithm erroneously coding proper names as the names of locations where events took place.

The filters dictionary also performs other nuanced disambiguation tasks. A recurrent source of location error comes from local newspapers that have the state or municipality as part of their name (e.g. "*El Diario de Juárez*"). Another source of potential location error comes from news reports mentioning the registration state of vehicle license plates when they are seized by the authorities. For example, a report indicating that a vehicle with plates from state "X" is intercepted in location "Y" on its way to destination "Z" can cause confusion in the place detection protocol. The EVENTUS ID location filter dictionary helps to minimize the risk of coding error caused by these types of reports. Like the other dictionary files, the list of filters should be contained in a plain text file (*.txt*). The first element in each line must be a numeric code "0", followed by a tab, and then the text corresponding

to the exclusion category. Assigning the code "0" gives the program the instruction to ignore that match in the corpus of text.

The following list shows an example of the filters dictionary:

```
0 → Cartel de Sinaloa
0 → Cartel de Juarez
0 → Cartel de Tijuana
0 → 3/a Zona Militar La Paz BCS
0 → Operativo Conjunto Michoacan
0 → Operacion Conjunta Nuevo Leon
```

4.2 Event Location Using Eventus ID

To identify the location of an event, Eventus ID combines the following five files. Users can refer to the description of these files in Section 3.1.

- corpus_DEMO.txt is the corpus of text to be analyzed.
- codigos_Events_DEMO.txt is the output file of numerical event codes generated by the event coding procedure.
- Location dictionaries: lists of places and their codes.
  - states_DEMO.txt is the dictionary containing the list of states.
  - municipalities_DEMO.txt contains the list of municipalities and towns.
- filters_DEMO.txt is a set of items for disambiguating locations.

The event location coding procedure is outlined in Table 4.1. In general, the location algorithm uses the event database generated by the event coding algorithms and identifies the source paragraph from which each event was extracted. It then reads the entire text corpus in order to identify the specific paragraph containing that event. Once the paragraph is identified, the algorithm uses the information provided by the location dictionaries to search for the name of a state or municipality in the paragraph. If a location is identified, the protocol uses the filters dictionary to verify whether the location should be assigned or discarded. If the location is not filtered, the algorithm saves the location code next to the corresponding event in the event data set. If no location is found in the paragraph, the algorithm expands the search to the rest of the document (Chapter 2 discusses the nomenclature and formatting characteristics of the text corpus that enable all paragraphs that constitute a single document to be identified). If a state or municipality is recognized in the corpus, the protocol checks whether it should be filtered or not. If it passes the filter, the

algorithm saves the code of the location next to its corresponding event code in the event database. The coding protocol is capable of saving the names of multiple localities as they are mentioned in the paragraph. If no location is identified in the document, the protocol stops searching for the location of this event and moves to the next episode in the event coding database.

Table 4.1: EVENTUS ID event location algorithm

1) Identify an event
- Load the event database. - Select an event from a new line. - Identify the paragraph (`FileName_P1_P2`) from which the event was extracted. - Use the entire paragraph name as searching criteria.
2) Identify the paragraph in the text corpus.
- Load the text corpus. - Search for the paragraph from which the event was extracted.
3) Search for the location of the event in the paragraph.
- Load the location dictionaries (states and municipalities). - Use the items of the location dictionaries as searching criteria. - Start searching for the location in the source paragraph. - If the location is found, store the code. - Keep searching for locations and storing them until the end of the paragraph. - If there are no more locations in the paragraph, then go to Step 5. - If no location is found in the paragraph, go to Step 4.
4) Expand the search to the rest of the document.
- Select the remaining paragraphs belonging to the same document (`FileName`). - Search for the location in all paragraphs of the document. - Begin searching in the first paragraph. - If the location is found in the document, store the location code and go to Step 5. - If the location is not found in the document, stop searching and go to Step 1.
5) Filter the location.
- Load the filters dictionary. - Verify that the location identified does not match any item in the filters dictionary. - If the location matches a filter, go back to Step 3. - If the location does not match a filter, go to Step 6.
6) Save the location.
- Save the location at the end of the coded event line in the event database. - Start again from Step 1.

For example, consider the following paragraph:

English:

`In a press release issued yesterday, the Mexican government announced that troops deployed in the municipality of San Luis Rio Colorado, Son. seized 227 packages of marijuana with a total weight of two tonnes and 250 kilograms while patrolling rural roads in the area.`

Spanish:

```
En un comunicado de prensa emitido el dia de ayer, el gobierno mexicano
informo que tropas destacamentadas en el municipio de San Luis Rio Col-
orado, Son. decomisaron paquetes de mariguana con un peso total de dos
toneladas y 250 gramos, mientras patrullaban caminos rurales del area.
```

Given the right set of actors, verbs and location dictionaries, EVENTUS ID will be capable of recognizing the key components of the event as "`troops`" (source), "`seized`" (action) and "`packages of marijuana`" (target). In addition, the location algorithm will be able to identify the municipality of "`San Luis Rio Colorado`" in the state of "`Sonora`" (Son.) as the location where the event took place.

4.3 Georeferenced Event Data

The location algorithm makes use of the numerical or textual output files generated at the event coding stage and, if a location is assigned, adds the coded location information. Depending on the option selected by the user in the location coding configuration (see Step 12 in Section 3.1.1), the event location will generate an output file in plain text format (*.txt*) containing the numeric codes of locations (codigos_Geo_DEMO.txt), the corresponding text (textos_Geo_DEMO.txt) or both files. The final output of this procedure is a file containing the source, action, target, date and location of an event. In this way, EVENTUS ID provides integrated event information on who did what to whom, when and where.

The following example illustrates the content of the output file of georeferenced events indicating the state and municipality codes. The entries `actor1`, `verb`, `actor2`, `state1` and `mun1` can be numerical codes or textual annotations depending on the output file generated by EVENTUS ID (see Section 3.1).

```
date1 FileName_P1_P1  actor1 verb  actor2 Edos   state1 Muns   mun1
date1 FileName_P2_P2  actor1 verb  actor2 Edos_i state1 Muns_i mun1
date2 FileName_P1_P3  actor1 verb  actor2 Edos   state1 Muns   mun1
date3 FileName_P1_P4  actor1 verb  actor2 Edos   state1 Muns   mun1
date3 FileName_P2_P5  actor1 verb  actor2 Edos_i state1 Muns_i mun1
date3 FileName_P3_P6  actor1 verb  actor2 Edos_i state1 Muns_i mun1
date4 FileName_P1_P7  actor1 verb  actor2 Edos   state1 Muns   mun1
```

4.4 Imputed events

As the previous example shows, the output contains two labels indicating the information that corresponds to the geographic location of events. The delimiter "`Edos`" marks the begining of state codes and the lable "`Mun`" indicates the municipality codes. The output can contain as many state and municipality codes as mentioned in the paragraph of press release under analysis. For simplicity, this example considers only one state and municipality code.

According to the event location algorithm (see Steps 3 and 4 in Table 4.1), EVENTUS ID is capable of identifying the place of occurrence of an event even when such information is not contained in the specific paragraph being coded. To do so, the program gathers information from other parts of the news report and imputes the name of the locality when it is found elsewhere in the document. As it is often the case, press releases and news reports usually indicate the location of the event in the header or the leading line, and then describe the episode in further detail without mentioning the event locality again. To reduce the risk of missing valuable georeferenced data, this feature relies on pieces of information contained in the broader document to impute the event location. The program only extends the location search up to the document level when no georeferencing information is identified at the paragraph level. However, the location search is not extended to the rest of the corpus. The delimiter labels "`Edos_i`" and "`Mun_i`" in the output file indicate the state and municipality codes that were imputed from other paragraphs of the document.

Chapter 5

Validation and Recoding

Measurement validity is a central concern for social scientists whether they use quantitative or qualitative methods (King, Keohane and Verba, 1994; Adcock and Collier, 2001; Collier and Brady, 2004; Bollen, 1989; Goertz, 2005). Measurement validity is achieved when the scores meaningfully capture the ideas contained in the corresponding concepts. Simply stated by Bollen (1989, 184), a score is valid if "a variable measures what it is supposed to measure." According to Adcock and Collier (2001), measurement validity should be understood in relation to the congruency between concepts and observations. These authors propose an analytical framework for evaluating measurement validity in terms of the degree of congruency at four levels:

1. Background concept. This level encompasses the constellation of diverse meanings associated with a given concept.
2. Systematized concept. This level contains the specific formulation or definition of a concept adopted by a particular researcher.
3. Indicators. This level refers to the procedure used for systematically building the measures associated with the definition.
4. Scores for cases. This level refers to the numerical scores or qualitative classification assigned as values for each measure.

In general terms, the framework indicates that a measure is valid to the extent that the scores (level 4) correspond to a set of indicators (level 3), that can be meaningfully interpreted in terms of the definition (level 2) used to represent a broader concept (level 1).

EVENTUS ID carries out a score generation process primarily operating at the fourth level of the analytical framework. From a narrow point of view, the machine-generated database is valid to the extent that the dictionaries and the coding protocol accurately identify in the set of events relevant to the researcher in the text corpus. However, a broader validity criteria should take into consideration the alignment between the machine-generated data and the theoretical conceptualization conceived by the researcher, as well as the systematization of the concept and development of indicators. The following sections present recommendations for assessing the validity of the machine-generated event data from a narrow perspective (level 4) and leave the broader conceptualization effort (levels 1–3) up to the researcher.

5.1 Validation

Validate, validate, validate! This is the main recommendation proposed by Grimmer and Stewart (2013) and reemphasized in this manual. In general terms, the objective of the validation procedure is to reduce the risk of type I and II errors. Users may reduce the risk of false negatives (type II error) by developing a detailed, comprehensive list of actors, verbs and locations capable of identifying the behavior of interest. In addition, users may develop a set of exceptions to the actor and verb dictionaries and location names to reduce the risk of false positives (type I error) that could lead to erroneously identifying an event that did not take place.

A basic validation method of event coding accuracy consists of three steps:

Step 1. Generate a human coding standard: First, select a random sample of documents or paragraphs from the corpus of text. Next, have a team of human coders identify event data in the sample corpus. Coders must follow the steps of the event and location algorithms implemented in EVENTUS ID (see Sections 3.4 and 4.2). The human-generated database serves as the golden standard for assessing the validity of the automated coding procedure.

Step 2. Compare human and machine coding: Use EVENTUS ID to generate an initial version of the event database. Then compare each item contained in the human-generated standard database with the computerized coding outcome. The discrepancies between the machine and human-generated database will help to identify the items that need to be modified in order to improve the quality of the automated coding scheme.

Step 3. Fine-tune the coding protocol: EVENTUS ID offers a wide degree of flexibility for users to generate coding protocols that best fit their research needs. Informed by the discrepancies between the human and computer-generated event data, users have these options available for improving the accuracy of the coding protocol:

- **Improve the actor and verb dictionaries**: The most common way to improve the accuracy of the coding protocol is by fine-tuning the dictionaries of actors and verbs. This usually requires the inclusion of additional nouns and verbs not contained in the previous versions of the dictionaries. Users may also need to use the null code [- - -] to disambiguate specific cases.

- **Consider using two actor dictionaries**: EVENTUS ID offers the possibility of using different actor dictionaries to identify the source and target of events. When doing so, users should bear in mind the sequence of actor identification implemented by the general and partial coding algorithms, as well as the quality of the source text.

- **Select the appropriate event coding algorithm**: The program gives researchers the possibility to code events using only the general sequence procedure or also using the partial sequence algorithm. Researchers may assess the accuracy of the event identification output generated with one or both coding algorithms and determine which option works best for the project at hand.

- **Improve location dictionaries and filters**: Identifying the location of an event can be a challenging task. Users may need to enhance and refine their state and municipality dictionaries in order to increase the probability of identifying a match in the source text. In addition, and perhaps most importantly, users may need to develop a robust set of location filters to avoid the risk of erroneously identifying some nouns as event localities.

As shown in Figure 1.1 in Chapter 1, the validation process consists of a cycle of several iterations of coding, validation, tuning and recoding and so on. After the first coding outcome, the user evaluates the discrepancies between human and machine coding and uses the information to improve the coding protocol. This learning is used for a second round of machine coding, which is then evaluated against the human standard and used to inform further modifications to the coding protocol. Carrying out several iterations of this process will increase convergence between the human coding and the machine-generated database. However, researchers must be aware that there is no 100 percent accurate coding protocol. Due to the complexities of natural language, it is extremely difficult to eliminate all discrepancies between human and computerized coding.

5.2 Accepting Defeat

Unfortunately, despite the effort of developing detailed dictionaries and taking advantage of the flexibility offered by EVENTUS ID, poorly written text may still

generate coding error. Since the quality of text is generally outside the control of the researcher, users may have no other option than to accept defeat and report a quantitative measure of coding error. For example, consider the following excerpt from a real news report (Proceso, 2014):

Spanish:

`El Grupo de Coordinacion Tamaulipas, que se integra con fuerzas de seguridad estatales y federales, comunica que tres civiles armados perdieron la vida en el municipio de Matamoros, luego de que atacaron a elementos de la Secretaroa de Marina que realizaban labores de vigilancia en helicoptero, quienes los abatieron tras repeler la agresion.`

English:

`The Tamaulipas Coordination Group, integrated by federal and state security forces, announces that three armed civilians lost their lives in the Matamoros municipality after they attacked Navy troops conducting surveillance activities in a helicopter, who cut them down after repelling the attack.`

Human coders would be likely to extract the following three events from this paragraph:

`armed civilians` → `attacked` → `Navy troops`
`Navy troops` → `repelled` → `armed civilians`
`Navy troops` → `killed` → `three armed civilians`

The syntactic structure in this paragraph is so complicated (if not cluttered) that the event coding algorithms of EVENTUS ID would be unlikely to be capable of extracting these three events; that is, of generating an accurate description of who did what to whom. Unfortunately, sometimes the poor quality of the original text defeats the researcher's ability and efforts.

5.3 Recoding

After generating a validated database of event data using EVENTUS ID, some users might wish to carry out an additional recoding to improve the accuracy of the database. We have not developed a recoding function embedded in EVENTUS ID, but manipulating event data should be straightforward with statistical programs such as STATA or R. The demonstration file (Eventus_ID_DEMO.zip) contains a set of STATA dofile examples for recoding and aggregating the raw event data generated by EVENTUS ID.

Data recoding might be particularly necessary for making sense of events extracted from passive voice sentences. As noted in Chapter 3, the use of passive voice is quite common in news reports written in Spanish. Sentences using the passive voice form invert the order of the subject and object in the syntactic structure, which might lead to misinterpretation of the coded event when analyzed out of its content. For example, consider the following sentence in passive voice:

`A member of a criminal group was arrested by Army troops.`

Given the structure of the sentence, EVENTUS ID will code this event as:

Textual: `member of a criminal group` → `was arrested` → `Army troops`

Numeric: `601060` → `99104` → `202051`

Syntactically, the sentence presents first the object, then the verb (in past simple sith a passive construction) and finally the subject. Given this structure, the coding protocol will identify first "member of a criminal group", then the verb "arrested" and finally "Army troops." However, by focusing exclusively on the numeric codes (as the statistical analysis software would do), the codification would not be able to accurately reflect the directionality of the event as indicated in the paragraph. In fact, the decontextualized reading of the numeric code might wrongly suggest that "a member of a criminal group" (601060) "arrested" (99104) "Army troops (202051)," which is not the idea presented in the paragraph.

In this case, users might develop a recoding protocol to reverse the object-verb-subject structure in a passive sentence and generate a subject-verb-object syntactic structure. To facilitate the task of recoding events extracted from passive sentences, we suggest including the prefix `99` before the actual verb code (see Section 3.3). This prefix helps users to identify grammatical structures in passive voice and correct the event directionality as follows:

Textual: `Army troops` → `arrested` → `member of a criminal group`

Numeric: `202051` → `99104` → `601060`

5.4 Aggregation and Duplicates

EVENTUS ID is designed for identifying discrete events mentioned in the text source; specifically detecting the source, action, target, date and location of each event. Given the temporal and geographic distribution of events, the output file is likely to generate a list of discreet event codes scattered across time and space. This type of information is not readily structured in a regular time-series cross-sectional structure for conducting statistical analysis or data visualization. To conduct quantitative analysis of the output generated by EVENTUS ID, users must aggregate the data into regular intervals of time (T) (e.g. year, month, day) and spatial units (N) (e.g. state or municipal level) to make up a panel data structure ($T \times N$).

STATA offers straightforward procedures such as the `collapse` command for aggregating data into different spatial and temporal units (see `http://www.stata.com/help.cgi?collapse`). Depending on their specific research objectives, users might use this command to aggregate the data by calculating the counts, raw sums or averages of event data.

It is important to note that users employing multiple sources of information are likely to obtain multiple reports of some events. As mentioned by Davenport and Ball (2002) and Davenport (2009), the media tends to over-report highly prominent events while other instances of lesser prominence usually receive scant media attention. Besides multiple independent reports of the same episode, another source of artificial event inflation might come from situations being described several times within the same news report. In such cases, repetition and redundant information can cause EVENTUS ID to identify the same event multiple times.

To avoid the risk of artificially inflating events due to over-reporting, users should consider eliminating multiple coded events. The usual recommendation is to keep only one event per day at the smallest spatial unit of analysis, and eliminate duplicate records on that same unit-day. The STATA command `duplicates` provides an easy way of reporting, obtaining examples, listing, browsing, tagging and eliminating duplicate observations (for details see `http://www.stata.com/support/faqs/data-management/duplicate-observations/`).

Chapter 6

Special Coding Projects

6.1 Identifying the Location of Actors

EVENTUS ID has the flexibility to be adapted to a wide variety of research objectives. In some cases, users might be interested in special coding projects focused on detecting particular pieces of information without taking full advantage of the general and partial event coding algorithms nor the two-level location capabilities of EVENTUS ID. For example, consider a research project in which the researcher aims to identify the presence of specific actors at the state level, without distinguishing their actors nor their distribution at the municipal level.

To carry out a project of this type, users can use the blank_DEMO.txt file as a wild-card to cause EVENTUS ID to ignore some searching criteria. The blank file blank_DEMO.txt is an empty document in plain text format (*.txt*) that can be used as a substitute for any combination of actors, verbs and location dictionaries in EVENTUS ID. Since the file contains no information, it provides no categories for the program to search for in the text corpus.

In order to design a coding protocol for the project described in the above example, the user could use a configuration file config_DEMO.txt with the following content:

```
actors_DEMO.txt
blank_DEMO.txt
blank_DEMO.txt
corpus_DEMO.txt
Events_DEMO.txt
blank_DEMO.txt
```

```
states_DEMO.txt
filters_DEMO.txt
Geo_DEMO.txt
3
```

The first entry instructs EVENTUS ID to use the actor dictionary to identify the source of the event. The second and third entries use the blank file to prevent the program from searching for actions or targets. The fourth item points to the text corpus, and the fifth item provides the name for the event coding output file. In the sixth line, the blank file causes municipal location codes to be ignored. The seventh item uses the states dictionary to enable the location of the actors to be identified at the state level. The eight item enables the use of filters for disambiguation of places, and the ninth item indicates the name of the geo-referenced outcome file. The tenth line instructs the program to generate both numeric and textual versions of the output file. In this way, the user can take advantage of the coding flexibility of EVENTUS ID to generate a custom coding protocol that identifies actors at the state level while ignoring other information.

6.2 Locating Specific Behaviors

As another example, if users are primarily interested in identifying specific actions without being concerned about who the perpetrators or the targets, then they can use the following config_DEMO.txt file:

```
blank_DEMO.txt
verbs_DEMO.txt
blank_DEMO.txt
corpus_DEMO.txt
Events_DEMO.txt
blank_DEMO.txt
states_DEMO.txt
filters_DEMO.txt
Geo_DEMO.txt
3
```

In this example, the configuration uses the blank file to ignore the source and target actors and focus specifically on identifying the actions in the list of verbs. The configuration file also prevents EVENTUS ID from searching for localities at the municipal level. The outcome of this special coding structure would be a series of specific actions identified at the state level.

Appendix A

Version Release History

A.1 Version 1.0, (beta) (May 2013)

Initial development of EVENTUS ID features. Version not public.

Works on Unix operating systems.
Corpus formatted in 80-character length limit.
General and partial event coding algorithms.
Event location facility works independently.
Basic interface.

A.2 Version 2.0 (June 2014)

First public version of EVENTUS ID.

Adapted to run on Windows operating system.
Corpus formatted in single line with no length limit.
Refinement of general and partial event coding algorithms.
Event location facility is fully integrated into the program.
Interface improvement.
Enable configuration file (config.txt).

Bibliography

Adcock, Robert and David Collier. 2001. "Measurement Validity: A Shared Standard for Qualitative and Quantitative Research." *The American Political Science Review* 95(3):529–546.

Alcoba Rueda, Santiago. 1983. "El presente de los titulares de prensa: no deíctico, protiempo anafófico.".
URL: *http://dfe.uab.es/dfeblog/salcoba/files/2008/10/presente_titulares_tiempo_anafora.pdf*

Baumgartner, Frank, Bryan Jones and Michael MacLeod. 1998. "Lessons from the Trenches: Ensuring Quality, Reliability, and Usability in the Creation of a New Data Source." *The Political Methodologist* 8(2):1–10.

Beieler, John. 2013. "Mapping Protest Data.". Last accessed on 4/24/2014.
URL: *http://johnbeieler.org/blog/2013/07/03/mapping-protest-data/*

Bigert, Johnny. 2005. Automatic and Unsupervised Methods in Natural Language Processing PhD thesis KTH Royal Institute of Technology.
URL: *http://www.diva-portal.org/smash/get/diva2:7478/FULLTEXT01.pdf*

Bollen, Kenneth A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Collier, David and Henry E Brady. 2004. *Rethinking Social Inquiry: Diverse Tools, Shared Standards*. Maryland: Rowman & Littlefield Publishers.

Coppedge, Michael and H. Reinicke Wolfgang. 1990. "Measuring Polyarchy." *Studies in Comparative International Development* 25(1):51–72.

Davenport, Christian. 2009. *Media Bias, Perspective, and State Repression: The Black Panther Party*. New York: Cambridge University Press.

Davenport, Christian and Patrick Ball. 2002. "Views to a kill: exploring the implications of source selection in the case of Guatemalan State Terror, 1977-1995." *Journal of Conflict Resolution* 46(3):427–450.

Geddes, Barbara. 2003. *Paradigms and Sand Castles: Theory Building and Research Design in Comparative Politics*. Ann Arbor, MI: University of Michigan Press.

Goertz, Gary. 2005. *Social Science Concepts: A User's Guide*. Princeton, New Jersey: Princeton University Press.

Grimmer, Justin and Brandon M. Stewart. 2013. "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* Published.
URL: *http://pan.oxfordjournals.org/content/early/2013/01/21/pan.mps028.abstract*

Guízar García, Elizabeth. 2004. El uso de los verbos en los titulares de cinco diarios de la ciudad de México: análisis sintáctico PhD thesis Universidad Nacional Autónoma de México Mexico: .

Hanna, Alex. 2014. "Assessing GDELT with handcoded protest data.". Last accessed on 4/24/2014.
URL: *http://badhessian.org/2014/02/assessing-gdelt-with-handcoded-protest-data/*

Harvey, Anna. 2008. What Makes a Judgment "Liberal"? Coding Bias in the United States Supreme Court Judicial Database. In *3rd Annual Conference on Empirical Legal Studies Papers*.
URL: *http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1120970*

Instituto Nacional de Estadística y Geografía. 2011. "Marco Geoestadístico Nacional.".
URL: *http://www.inegi.org.mx/geo/contenidos/geoestadistica/default.aspx*

King, Gary, Robert O. Keohane and Sidney Verba. 1994. *Designing Social Inquiry*. Princeton University Press.

Leetaru, Kalev and Philip A. Schrodt. 2013. "GDELT: Global Data on Events, Location and Tone, 1979-2012.".

Lewis, M. Paul, Gary F. Simons and Charles D. Fennig. 2013. "Summary by language size.". Last accessed on 6/10/2014.
URL: *http://www.ethnologue.com/statistics/size*

Martínez, Francisco, Lucas Miguel and Cristian Vázquez. 2004. "La titulación en la prensa gráfica.".
URL: *http://www.perio.unlp.edu.ar/grafica1/htmls/apuntescatedra/apunte_titulacion.pdf*

Moore, Will H. 2014*a*. "No More Fountains of Youth/Pots o' Gold: Conceptualization and Events Data (Part 1).". Last accessed on 4/24/2014.
URL: *http://willopines.wordpress.com/2014/03/03/no-more-fountains-of-youthpots-o-gold-conceptualization-and-events-data-part-1/*

Moore, Will H. 2014*b*. "No More Fountains of Youth/Pots o' Gold: Conceptualization and Events Data (Part 2).". Last accessed on 4/24/2014.
URL: *http://willopines.wordpress.com/2014/03/04/no-more-fountains-of-youthpots-o-gold-conceptualization-and-events-data-part-2/*

Nadal Palazón, Juan. 2009. *El Discurso Ajeno en los Titulares de la Prensas Mexicana*. Mexico City: Universidad Nacional Autónoma de México.

Open Event Data Alliance. 2014. "Objectives - Open Event Data Alliance.". Last accessed on 6/10/2014.
URL: *http://openeventdata.org/*

Osorio, Javier. 2013. "Hobbes on Drugs: Understanding Drug Violence in Mexico.".

Osorio, Javier and Alejandro Reyes. 2014*a*. "Web 2 Eventus.".
URL: *http://www.javierosorio.net/#!software/cqbi*

Osorio, Javier and Alejandro Reyes. 2014*b*. "Web Text Downloader.".
URL: *http://www.javierosorio.net/#!software/cqbi*

Price, Megan and Anita Gohdes. 2014. "Searching for Trends: Analyzing Patterns in Conflict Violence Data.". Last accessed on 4/24/2014.
URL: *http://politicalviolenceataglance.org/2014/04/02/searching-for-trends-analyzing-patterns-in-conflict-violence-data/*

Proceso, Redacción. 2014. "Abaten a tres sicarios que atacaron helicóptero de la Semar.".
URL: *http://www.proceso.com.mx/?p=370045*

Rodríguez, Juan Manuel. 2001. "Errrores cumunes en el lenguaje periodístico: Invasión Pasiva." *Revista Latinoamericana de Comunicación CHASQUI* .
URL: *http://www.redalyc.org/pdf/160/16007609.pdf*

Rohner, Ronald P. and Leonard Katz. 1970. "Testing for Validity and Reliability in Cross-Cultural Research." *American Anthropoligist* 72:1068–1073.

Schrodt, Philip A. 2009. "TABARI. Textual Analysis by Augmented Replacement Instructions.".

Schrodt, Philip A. 2014. "The legal status of event data.". Last accessed on 6/10/2014.
URL: *http://asecondmouse.wordpress.com/2014/02/14/the-legal-status-of-event-data/*

Schrodt, Philip A. and Deborah Gerner. 1994. "Validity Assessment of a Machine-Coded Event Data Set for the Middle East, 1982-1992." *American Journal of Political Science* 38:825–854.

Schrodt, Philip A. and Deborah Gerner. 2012. Fundamentals of Machine Coding. In *Analyzing International Event Data: A Handbook of Computer-Based Techniques.*
URL: *http://eventdata.psu.edu/papers.dir/AIED.2012.ch2.pdf*

Sipes, R. G. 1976. "A Test For Coder Bias." *Cross-Cultural Research* 11(3):149–168.

Steinert-Threlkeld, Zachary. 2014. "Machine Coded Events Data and Hand-Coded Data.". Last accessed on 4/24/2014.
URL: *http://politicalviolenceataglance.org/2014/03/19/machine-coded-events-data-and-hand-coded-data/*

Tversky, A and D Kahneman. 1974. "Judgment under Uncertainty: Heuristics and Biases." *Science (New York, N.Y.)* 185(4157):1124–31.

Ulfedler, Jay. 2013*a*. "Road-Testing GDELT as a Resource for Monitoring Atrocities.". Last accessed on 4/24/2014.
URL: *https://dartthrowingchimp.wordpress.com/2013/05/02/road-testing-gdelt-as-a-resource-for-monitoring-atrocities/*

Ulfedler, Jay. 2013*b*. "The Future of Political Science Just Showed Up.". Last accessed on 4/24/2014.
URL: *https://dartthrowingchimp.wordpress.com/2013/04/10/the-future-of-political-science-just-showed-up/*

Ward, Michael, Andreas Beger, Josh Cutler, Matthew Dickenson, Cassy Dorff and Ben Radford. 2013. "Comparing GDELT and ICEWS Event Data.". Last accessed on 4/24/2014.
URL: *http://mdwardlab.com/sites/default/files/GDELTICEWS_0.pdf*

Weller, Nicholas and Kenneth McCubbins. 2014. "Raining on the Parade: Some Cautions Regarding the Global Database of Events, Language and Tone Dataset.". Last accessed on 4/24/2014.
URL: *http://politicalviolenceataglance.org/2014/02/20/raining-on-the-parade-some-cautions-regarding-the-global-database-of-events-language-and-tone-dataset/*