AUTOMATED CODING OF INTERNATIONAL EVENT DATA USING SPARSE PARSING TECHNIQUES

Philip A. Schrodt [*]

University of Kansas

"Event data" record the interactions of political actors reported in sources such as newspapers and news services; this type of data is widely used in research in international relations. Over the past ten years, there has been a shift from coding event data by humans—typically university students—to using computerized coding. The automated methods are dramatically faster, enabling data sets to be coded in real time, and provide far greater transparency and consistency than human coding. This paper reviews the experience of the Kansas Event Data System (KEDS) project in developing automated coding using "sparse parsing" machine coding methods, discusses a number of design decisions that were made in creating the program, and assesses features that would improve the effectiveness of these programs.

Key words: event data, natural language, sparse parsing, international relations, social science

Paper presented at the annual meeting of the International Studies Association, Chicago, February 2001. An earlier version of this paper was presented at the Fifth International Conference on Social Science Methodology, Cologne, October 2000.

[*] Address: Department of Political Science, University of Kansas, Lawrence, KS 60045, USA.
Email: p-schrodt@ukans.edu.

1 INTRODUCTION

Event data are used in quantitative international relations research to reduce journalistic descriptions of political interactions to categorical data that can be analyzed statistically. They are one of the most common types of information used in quantitative international relations research (Andriole and Hopple 1984; Azar and Ben-Dak 1975; Burgess and Lawton 1972; Daly and Andriole, 1980; Laurance 1990; Marlin-Bennett and Roberts 1993; Merritt, Muncaster and Zinnes 1994; Munton, 1978; Schrodt 1994; Sigler, Field and Adelman 1972)

An early paper by McClelland (1967) provides a general definition of international events:

> Event-interaction is meant to refer to something very discrete and simple—to the veritable building blocks of international politics. They are the specific elements of streams of exchange between nations. Here are a few examples for hypothetical Nations A and B: Nation A proposes a trade negotiation, Nation B rejects the proposal, Nation A accuses B of hostile intentions, Nation B denies the accusation, Nation B deploys troops along a disputed boundary, Nation A requests that the troops be withdrawn, ... Each act undertaken by each actor in this illustration is regarded as an event-interaction. (pg. 8)

Two widely-used global event data sets were developed in the 1970s—the Conflict and Peace Data Bank (COPDAB; Azar 1982) and the World Events Interactions Survey (WEIS; McClelland 1976)—and a variety of more specialized data sets covering specific regions or time periods are also available.

Historically, event data have been coded by teams of undergraduate and graduate students reading through copies of the *New York Times* and other printed sources. During the past decade, however, there was a shift to automated coding methods: In 1990 almost all event data projects used human coders, whereas in 2000 almost all projects used automated coding. Projects that once would have required tens of thousands of dollars, a large group of coders with a complex supervisory infrastructure and months of painstaking effort can now be done by a single researcher in a few days or weeks.

This change is particularly noticeable in the research published in major political science journals such as the *American Political Science Review* (Goldstein and Pevehouse 1997; Schrodt and Gerner 2000), *American Journal of Political Science* (Schrodt and Gerner 1994) and *Journal* *of Conflict Resolution* (Schrodt and Gerner 1997; Bond et al 1997). Huxtable (1997) and Thomas (1999) are examples of completed dissertations based on large-scale event data sets generated by a single researcher using fully automated methods, and several other dissertations are in progress. Human-coded data has continued to be employed in some government-sponsored projects, most notably the State Failures Project (Esty et al 1995, 1998), but even these efforts are gradually moving to automated coding.[1]

Improvements in communication technologies also have dramatically changed the quantity and timeliness of the information available for use in event data analysis. Machine-readable text relevant to political behavior is now available the commercial efforts of Reuters, *Agence France Press*, and other news agencies; reports from hundreds of news sources can be accessed at most academic institutions through the NEXIS "Academic Universe" program.

Machine coding provides at least two advantages over traditional human coding. First, as Weber (1990:17) notes, coding reliability in textual analysis consists of three components:

> ***stability***—the ability of a coder to consistently assign the same code to a given text;
> ***reproducibility***—intercoder reliability;
> ***accuracy***—the ability of a group of coders to conform to a standard.

The stability of machine coding is 100% because the machine will always code the same text in the same manner. This is particularly useful when a time series is being maintained for a number of years. Because the patterns used in coding are explicitly specified in the coding dictionaries rather than dependent on coder training, the same rules used in 1992 can be used in 2002. In our experience, the reproducibility of machine coding seems comparable to the inter-coder reliability of humans, and a machine is obviously not influenced by the context of an event or by intrinsic political or cultural biases. (The dictionaries used by a coding program may reflect biases, but these will be explicit and can be examined by another researcher; the dictionaries are also applied consistently to all actors and in all contexts.) Automated coding is not subject to errors due to fatigue or boredom, and in contrast to human coders, a computer program does not require retraining after a coding vocabulary has been developed.

[1] The last major international relations event data project using human coding—the Global Event Data System (GEDS) project at the University of Maryland—abandoned that approach in December 2000 and is now working on a fully-automated system that will use TABARI.

Second, automated coding can be done quickly and inexpensively. This allows a researcher to experiment with a variety of different coding schemes, which is impractical with human coding. For example, the Protocol for the Assessment of Nonviolent Direct Action (PANDA) at the Program on Nonviolent Sanctions in Conflict and Defense at the Center for International Affairs at Harvard (Bond, Bennett, and Vogele 1994) used the KEDS program to code a superset of the WEIS categories (160 categories versus the 63 categories in WEIS) that provided far more detail on nonviolent events, substate actors and internal interactions such as strikes and protests.

2 THE KANSAS EVENT DATA SYSTEM PROJECT

The Kansas Event Data System (KEDS) was the first large-scale automated event coding effort in the academic community and remains the only system that is widely used in published academic research in political science. The project began in the late 1980s with funding from the "Data Development in International Relations" project of the U.S. National Science Foundation and has been continued under a variety of NSF and U.S. government contracts. The project web site—`http://www.ukans.edu/~keds`—contains the software and source code produced by the project, data sets, research papers and links to related projects. The KEDS project has been a collective effort involving several faculty and graduate students—most notably Deborah Gerner, Phillip Huxtable, Jon Pevehouse and Judith Weddle—and in this discussion I will use the first-person plural pronoun "we" to refer to the joint experience of this group.

The KEDS project initially developed an eponymously named automated coding program that was written in the Pascal computer language and worked only on the Macintosh operating system. While the source code was available after 1997, that code had not been intended for public release and consequently was not particularly well documented or easy to follow. In addition, while the program continued to be compatible with new versions of the Macintosh operating system, by the late 1990s the Pascal language had been almost completely superceded by the C/C++ language in most applications.

In 2000, we developed a new coder, TABARI ("Textual Analysis by Augmented Replacement Instructions"). TABARI is based on the same natural language processing principles as KEDS, but is designed to be more easily extended, and also corrects some dysfunctional but deeply-embedded design characteristics of KEDS. TABARI is written in ANSI-standard C++ and therefore is easily transferred between computer operating systems.

The system contains no proprietary components or code that is likely to become outdated due to operating system dependencies. (The disadvantage of this flexibility is that the user interface is a keyboard driven “teletype” interface, in contrast to the graphical interface of KEDS.) The reference operating system for TABARI is Linux, but it has also been ported to the Solaris, Macintosh, and Windows operating systems. It is “open-source” software, which means that not only the program but also the source code are available for review and modification by the research community.

TABARI is about 70-times faster than KEDS. Using an automatic coding mode that provides no screen feedback, TABARI codes 2000 events per second on a 350Mhz Macintosh G3 computer; on a 650Mhz Dell Pentium III, the speed is around 3000 events per second. Typical human coders can reliably produce about 40 events per day, so TABARI running on a G3 does in one second what a human coder does in about three months. This is wall-clock speedup of around a factor of 7.8-million.

We have experimented with coding a variety of texts, including specialized regional sources in English and German (Gerner et al 1994). Most of our initial work was with Reuters News Service lead sentences. The lead is usually a simple declarative sentence that summarizes the article, e.g., "`The United Arab Emirates welcomed a resumption of formal diplomatic ties between Egypt and Syria after a 12-year rift.`" Reuters articles are quite short and a complex political interaction generates dozens of leads. In our later work, the system has been employed successfully to code full newswire stories. In regions of the world that are less intensely reported than the Middle East—notably Africa—full story coding is essential; Schrodt and Gerner (1998) provide a system comparison of lead-sentence and full-story coding. KEDS’s coding on the Middle East has been validated against both the textual record and human-coded events and found no systematic biases in machine coding (Schrodt and Gerner 1994). When the human and machine-coded data are used in statistical tests, the results are almost indistinguishable except for differences due to the higher number of events in the machine-coded data. Our work at Kansas generated data primarily on the Middle East, with some additional work by graduate students focusing on West Africa and the Horn of Africa. Research projects using KEDS at other institutions have looked at the Balkans, Central American, Korea, and Northern Ireland.

3 SPARSE PARSING

KEDS and TABARI use a computational method called "sparse parsing." Instead of trying to parse a sentence fully, the programs determine only the parts of a semtence required for event coding—for instance political actors, compound nouns and compound verb phrases, and the references of pronouns—and then employ a large set of noun and verb patterns to determine the appropriate actor and event codes. Unlike more complex full parsing, sparse parsing can be used successfully on unedited news wire text, uses simpler dictionaries, and is substantially faster than most full parsers. Less is more.

The task of machine coding of event data is simplified by the fact that event data are largely defined by sets of transitive verbs (verbs that have a direct object). In most cases, event data coding needs only to focus on the basic subject-verb-object (SVO) structure of an English sentence, so event data coding is substantially simpler than many other natural language processing problems. In event coding, the subject of the sentence is the *source* of the event, the verb determines the *event code*, and the object of the verb is the *target*. Sparse parsing makes errors on oddly constructed or excessively complex sentences but it are successful on the sentence structures most commonly used to describe events.

3.1. How TABARI Evaluates a Sentence

The input to TABARI is a file containing a set of sentences, each prefixed with a date and other identifying information, and followed by a blank line:

```
980216  REUT-0001-01
Egypt's President Hosni Mubarak warned in an interview published on Monday
that the situation in the Arab world could deteriorate if the United
States attacks Iraq for failing to comply with weapons inspections.

980216  REUT-0002-01
Iraqi Foreign Minister Mohammed Saeed al-Sahaf said on Monday that he was
going to Paris only to take a message from President Saddam Hussein to
French President Jacques Chirac about Baghdad's showdown with the United
States.
```

(KEDS uses a two-digit year but “windows” the digits 00 to 10 to 2000-2010. TABARI can use either this system or 4-digit years.) While this format is very simple, we have found that the process of reformatting the text downloaded from a data service is often a difficult task for individuals unfamiliar with computer programming. Our web site contains a number of Pascal, C, and Perl programs that convert some of the formats we have used (for example NEXIS, Reuters, and Dow-Jones Interactive); other projects have written filters in languages such as Visual Basic and the Microsoft Word macro language.

To code a sentence, TABARI goes through a series of operations on the text. The following description is a summary and does not discuss a number of idiosyncratic exceptions and options employed by the parser.

Lexical processing (involves single words)

The source text is first converted to a standard form. All letters are changed to upper-case (however, words beginning with upper-case letters in mid-sentence are tagged as nouns); all punctuation except commas is eliminated. TABARI then checks each individual word in the text to see if it occurs in the actor or verb dictionaries, or in a list of conjunctions and pronouns. Words that are found in the dictionaries are assigned an integer identifier, and all subsequent processing works with this array of integers rather than the original text. The lexical level also assigns a word type to each literal. Most of TABARI's word types are conventional parts-of-speech—verb, pronoun, conjunction—but some are specialized, such as "actor", "number", and "comma."

Syntactic processing (involves multiple words)

The next step involves locating noun and verb phrases in the sentence. In many cases, these involve single words, but can also involve multiple words (e.g. "`President Bill Clinton`"). Some disambiguation of words that can be either nouns or verbs—for example "`force`", "`attack`", "`arms`" and "`strike`"—is done by looking for determiners ("`the`", "`a`" and "`an`") prior to the word and capitalization of words in mid-sentence. Pronoun references are determined using a simple set of rules; this is described below. Compound noun phrases, compound clauses and subordinate phrases delimited by commas are tagged; if a noun phrase does not occur at the beginning of a clause, it is copied from the beginning of the previous clause. Finally, a "complexity filter" is applied, and sentences that appear too complex for TABARI to code are written to a separate file rather than coded. At the end of this stage, the program generates a marked-up version of its parsing in an XML-like notation.

Event coding

To code events, the program looks at each verb in the sentence and attempts to match the phrases associated with that verb in the verbs dictionary. Patterns typically distinguish between direct objects, as in the distinction between "`promised military aid`" and "`promised to veto`."; each phrase is associated with an event code. If a verb phrase corresponding to an event is identified, the program finds the source actor and target actor associated with the verb. The source is usually the first actor in the sentence; the target is usually the first actor following the verb, provided that actor has a code distinct from the code of the source. If no such actor is found, the program then looks for an actor prior to the verb that has a code distinct from the code of the source. If the source or target are compound phrases, these are expanded into multiple events. Only the first verb corresponding to an event is coded, unless the sentence is compound (i.e. contains a conjunction not associated with a compound actor), in which case each clause of the compound sentence is checked for an event.

3.2 Dictionaries

TABARI was designed as a general-purpose coding system, so most of its coding decisions depend on its dictionaries. All of TABARI's files are stored externally as ASCII files so that they can be edited using a word processor and transmitted using electronic mail. The verbs and actors, as well as their associated codes, also can be added, deleted or changed using a dialog invoked from the main program menu as coding is being done. The editing routine also keeps track of the coder who added each phrase to the dictionary and the date the phrase was added.

A standard input format is used for the dictionaries. Words are entered in upper case; codes for actors and events are enclosed in square brackets []. If two words must be consecutive, they are connected by an underscore; if the two words are separated by a space, other words can intervene. For example, the text "`agreed to provide a loan`" can be matched by the pattern "`AGREE LOAN`" but not the pattern "`AGREED_TO_LOAN`".

If a word in a TABARI dictionary ends in a space, it is used as a stem. Stemming refers to the process of reducing different forms of a word to a single root

```
ACCEPT      ACCEPTS   ACCEPTED   ACCEPTING
SYRIA       SYRIA'S   SYRIAN     SYRIANS
```

TABARI handles stemming by matching patterns from the beginning of the word; a word is considered to match if every character in the root matches. In other words, "SYRI" will match all four forms of "Syria" but it will not match "Syracuse". Long phrases are searched before shorter ones: for example "SIGNALLED", "SIGNED" and "SIGN" are checked in that order. An underscore character after the word means that the word will match only if it is followed by a space, so the root "OF_" will only match the single word "of" whereas "OF" would match "of", "offer" and "official".

Stemming had two advantages in the early stages of our work with the original KEDS program when the dictionaries were relatively small. First, most regular verbs had to be entered only once. The exception to this occurred when a verb root could be mistaken for a noun—for example "FIRE"—in which case multiple forms of the verb had to be entered explicitly. Second, nouns related to a verb—for example "MEETING/MEET" or "ACCEPTANCE/ACCEPT"—would trigger a correct classification even when an uncommon verb in the text was not in the dictionary.

With more extensive dictionaries, however, stemming is the most frequent cause of wildly inaccurate coding errors. For example, when we were coding reports from the Middle East for November 1993, the usually problem-free verb "BEAT" matched "Beaty," the name of an American businessman released from prison by Iraq. TABARI would not make this mistake because it can disambiguate using capitalization, but at the present time TABARI still uses KEDS' stemming rules. Over the next year, we anticipate switching to a facility to TABARI for explicitly defining regular verb constructions and noun endings (e.g. plurals and adjectival forms) and eliminate stemming except for purposes of backwards-compatibility with KEDS.

The KEDS project coding dictionaries have been incrementally developed over the past decade through work at Kansas and in the PANDA project at Harvard. We now have a fairly stable set of about 4000 verbs and verb phrases that are sufficient to capture most of the political behavior coded in the WEIS event data scheme, and a standard list of about 500 major political actors. Earlier work (Lehnert and Sundheim 1991) on automated processing of English-language reports of political violence indicated that dictionaries on the order of 5,000 phrases are necessary for relatively complete discrimination between political events (including some activity more detailed than coded in WEIS), so these dictionaries are probably close to having a relatively complete vocabulary.

Proper nouns: actors

An actors file contains proper nouns and associates each of these with a code:

```
ABU_SHARIF [PLO]
ACQUINO [PHL]
AL-WAZIR [PAL]
AMMAN [JOR]
AMNESTY_INTERNATIONAL [NGO]
ANKARA [TUR]
ANTIGUA [ATI]
```

Multiple nouns can map to the same code; for example "ISRAEL", "NETANYAHU", "PERES", "RABIN", "TEL_AVIV", and "BARAK" all have the code ISR.

In some circumstances, it is useful to have a single "phrase" generate multiple actor codes. This is indicated by separating the actor codes with a slash:

```
EAST_AND_WEST_GERMANY [GME/GMW]
NORTH_AND_SOUTH_KOREA [KON/KOS]
G7 [USA/GMW/FRN/ITL/UK/JAP/CAN]
```

Actors also change over time. The two most notable instances of this in our Middle East data are Boutros Boutros Ghali—who appears both as Egypt's foreign minister and as Secretary General of the United Nations—and the changes surrounding the collapse of the Soviet Union. These situations are dealt with by indicating different codes for different time periods:

```
MOSCOW [USR (<901225) RUS (>901226)].
```

The standard actors dictionary contains the names of nation-states, major international organizations, and the heads-of-state of major powers. However, local political actors such as opposition parties, regional political leaders, rebel movements and local non-governmental organizations need to be customized for individual regions. We have developed a program called "Actor_Filter" that goes through a set of texts, pulls out potential new political actors (using capitalization to identify proper nouns), and then presents a keyword-in-context index of these, sorted by frequency. While this program produces quite a few false positives (for example, the names of holidays, athletes and movie stars), it insures that no frequently occurring political actor is missing from the dictionary.

Common nouns: agents[1]

The early event data sets such as WEIS and COPDAB were state-centered and made little or no distinction among substate actors. This convention is also found in many Reuters leads, where the names of the states are used to refer to actions of governments or foreign ministries: "`Israel accused Syria...`". However, in many circumstances it is useful to differentiate the agent responsible for an event, for example distinguishing "`Israeli soldiers,`" "`Israeli police,`" and "`Israeli settlers.`" This is particularly important in the PANDA coding scheme, which deals with many internal political activities such as strikes, elections, and protests.

If one is coding a small number of countries, the agents can be coded explicitly in the dictionaries:

```
ISRAELI_POLICE [ISRPOL]
ISRAELI_SETTLERS [ISRSET]
ISRAELI_SOLDIERS [ISRMIL]
```

While this approach leads to longer dictionaries, it also allows the secondary codes to be very specific: for example major opposition parties can be assigned distinct codes while minor parties are lumped together in a general "`OPP`" category.

The original KEDS program used in the PANDA project employed a more sophisticated approach was used that employed a single list of substate actors called "agents." (To date, this facility has not been implemented in TABARI.) In some cases, an agent is implicit in a proper noun—for example "`GEORGE BUSH`" was (and is again) president of the United States—and these codes are specified in the actors dictionary. In many other cases, the agent is identified by a common noun:

```
AGENT: DISSIDENT [OPP]
AGENT: ELECTORATE [CON]
AGENT: EMIGRANT [REF]
AGENT: EMIGRES [REF]
AGENT: ENVIRONMENTALIST [ENV]
```

KEDS attempts to assign an actor identity to all agents, using the following priority:

1. Implicit agents: `GEORGE BUSH [USA:GOV]`
2. <actor><agent>: `FRENCH POLICE`

[1] At the present time, agents are not implemented in TABARI, but will probably be added in the future.

3. <agent><preposition><actor>: POLICE IN DAMASCUS

4. an actor found within ±2 words of the agent

If none of these patterns occurs, the agent is assumed to have no explicit actor and is then treated as an actor when identifying sources and targets. The statement "Police fought demonstrators" will generate an event of the form "*** POL *** DEM" where *** is KEDS's code for an unknown actor.

Verbs and verb phrases

The verbs dictionary contains verb phrases and their associated event codes. This includes both simple verbs (e.g. "VISITED") and verbs plus direct objects (e.g. "PROMISED FUNDS").

The following is an example of an entry in the verbs dictionary:

```
ACCEPT
- * PROPOSAL     [081]
- PROPOSAL WAS * [081]
- * CHARGES      [013]
- * FORMULATION  [042]
- * INVITATION   [082]
```

In this example, the root verb is "ACCEPT"; with stemming this will match "ACCEPT", "ACCEPTS", and "ACCEPTED". The phrases that start with "-" are the patterns associated with "ACCEPT" and their codes; the "*" indicates where the verb itself should appear. In the example, "accepted proposal" will be coded 081 ("agree" in the WEIS coding system) while "accepts formulation" will be coded 042 ("approve" in WEIS).

When multiple verb patterns are found in a sentence, events are prioritized by:

• left to right order of verbs in the sentence

• length of patterns within a verb's pattern list

Events are coded from only the first verb in a sentence unless the sentence is compound or the first verb has been designated as "subordinate," a verb that is coded only if no other verb can be found to code.

Patterns usually involve direct objects or modifiers such as "NOT". The key to this scheme is ensuring that phrases are associated with a transitive verb rather than indicators of tense such as "HAS", "WILL", "IS", "WAS", and "WERE". The important verb in a phrase will often be an infinitive; for example in "willing to negotiate", the verb is "NEGOTIATE". Pattern matching stops at any conjunction; this prevents a pattern from matching the direct object of another verb in a compound sentence.

Patterns can also specify where the source and target are found in relation to the verb and associated words; these are indicated by "$" and "+" respectively. For example, the pattern

```
ADVISE
- + WAS * BY $
```

would make the correct source and target assignment on the passive construction `"Egypt was advised by the United States."` The symbol "%" specifies that a *compound actor* should be assigned to both the source and target. This is typically used when dealing with consultations to indicate that the subject of the sentence contains both the source and target.

```
Israeli businessmen, Jordanian officials and foreign bankers agreed on
Monday that the Israeli-Jordanian peace treaty was not producing economic
dividends quickly enough.
```

The current verbs dictionary requires relatively little modification when one is coding different regions of the world. This was not true initially: when we first tried to apply the dictionaries we had developed in the Middle East to the Balkans and the West Africa, we found that substantial additional vocabulary was required. Coding of unusual activities can also require specialized vocabulary. For example during the 1991 Gulf War, words that Reuters commonly used metaphorically to refer to verbal disputes (“`attacked`”, “`blasted`”) generally referred to actual physical attacks, and for precise coding, a different set of dictionaries would be required. The dictionaries required for coding internal political events (as distinct from international events) also require some regional customization, although the number of additional phrases required is usually in the tens or hundreds, not the thousands.

Pronouns

Pronouns occur frequently in Reuters:

```
Turkey believes Iraq and Syria can cope with a decrease in vital water but
they have lodged a protest with Ankara.
```

In this sentence, “`they`” refers to “`Iraq and Syria`” but the program must determine this in order to code the second clause of the compound sentence correctly.

Ascertaining the references of pronouns is a very general problem in parsing. Unfortunately, a reference often cannot be resolved on a purely syntactic basis. In the sentence “`Clinton will` `meet with Mubarak when he goes to Geneva`” the pronoun “`he`” could refer to either “`Clinton`” or “`Mubarak`” depending on who is going to Geneva. Sophisticated parsers (and humans) can often use semantic information to resolve references. In the sentence “`John took the cake home and ate it,`” the word “`it`” refers to “`cake`” because one does not eat “`home`”, whereas in the sentence “`John took the baseball bat and broke it`”, the word “`it`” refers to “`bat`” rather than “`baseball`.” Nonetheless, without very good disambiguation of parts of speech, the two sentences appear identical in structure.

TABARI does not use semantic information, but in most Reuters leads this is not required. Instead, a simple set of rules are used:

HE SHE IT	assign the first actor in the sentence
ITS	assign the first actor prior to the pronoun
THEY	assign either the first compound actor if one exists or else assign an actor followed by a word ending in 'S' or an agent, for example `“Syrian soldiers”` or `“Israel police.”`

These rules are least effective on the pronoun IT because that word often refers to an activity rather than an actor. For example in the sentence “`Police had said the rally was banned, but did not prevent it from taking place.`”, “`it`” refers to “`rally`.” In the sentence “`Interfax news agency, citing unnamed sources, said it would take four days for the troops to deploy`”, “`it`” has *no* reference, but instead serves as a placeholder for the implied subject of the sentence.

4 OTHER PARSING FEATURES

4.1 Compound Sentences

TABARI recognizes compound sentences generated by the conjunctions “`AND`” and “`BUT`”; this is determined after any “`AND`” found in compound a noun phrase such is “`Clinton and Mubarak`” is eliminated. In a compound sentence, the source of the event is not changed unless an actor occurs immediately after the conjunction.

TABARI recognizes compound nouns of the form

```
<actor$_1$> AND <actor$_2$>
<actor$_1$> , <actor$_2$> , ... <actor$_{n-1}$> AND <actor$_n$>
```

and does the appropriate duplication of events. For example, "The United States and Egypt approved of efforts by Israel and Jordan" would generate the four events

```
USA  <APPROVED>  ISR      USA  <APPROVED>  JOR
UAR  <APPROVED>  ISR      UAR  <APPROVED>  JOR
```

The consistent treatment of compound actors is one place where automated coding is far superior to human coding. One of the most common mistakes made by human coders is neglecting to complete all of the possible combinations of interactions in a sentence containing multiple compound nouns. Sparse parsing, in contrast, handles this consistently.

The most common error that results from these rules occurs when a compound noun is in the sentence but the nouns are not actors

> Teachers demanding *better pay and improved benefits* marched again through the streets of the center of Amman on Wednesday.

The “AND” will cause this to be interpreted as a compound sentence despite the fact that only the italicized clause is compound.

Conversely, one will occasionally find compound sentences where the phrases are separated with the compound noun structure

> The United States confirmed on Friday that it is prepared to respond positively to a U.N. appeal for more emergency food aid for *North Korea and U.S.* officials said the contribution would be $6 million.

In this instance, the second phrase in sentence—“ U.S. officials said…”—will not be interpreted as being compound because the italicized words will be parsed as a compound noun phrase.

4.2 Paired and Subordinate Codes

The WEIS coding scheme frequently generates symmetric events of the form

$$\text{<Actor}_1\text{> <Event}_1\text{> <Actor}_2\text{>}$$
$$\text{<Actor}_2\text{> <Event}_2\text{> <Actor}_1\text{>}$$

For example, a meeting between Israel and Egypt at a neutral site would generate the pair of events:

```
ISR 031 UAR         (meet with)
UAR 031 ISR         (meet with)
```

A visit by a Jordanian official to Syria would generate the event pair:

```
JOR 032 SYR         (visit; go to)
SYR 033 JOR         (receive visit; host)
```

In TABARI these paired events can be coded automatically by using a pair of codes separated by a slash ; for example

```
FLEW_TO [032/033]
```

would code the visit/receive pair.

TABARI dictionaries can also set priorities when multiple verbs are found in a sentence. A *subordinate* code indicates that a verb is only to be coded if no other events are found. When a phrase with a subordinate code is encountered, TABARI continues to search for other verb patterns in the sentence rather than stopping. This is typically used for verbs of attribution such as "`SAID`" or "`REPORTED`." For example in coding "`George Bush said he rejected Syria's assertion...`" the relevant event is `USA <REJECTED> SYRIA` rather than `USA <SAID> SYRIA.` Associating "`GEORGE_BUSH`" with the pronoun "`he`" and using a subordinate code on "`SAID`" will allows this to be coded properly.

4.3 Deletion of Subordinate Phrases Delimited by Commas

In Reuters leads, short phrases delimited by commas and phrases between a comma and the end of the sentence are usually irrelevant to the coding:

```
President Mubarak, in a grim warning underlining Egypt's deepening
economic crisis, will request emergency assistance from the IMF, the
official UAE news agency said on Thursday
```

These phrases are eliminated from the text to be coded if the number of words between the commas is greater than two and less than or equal to ten; the minimum allows the preservation of comma-delimited lists. The maximum and minimum length for an eliminated phrase can be set as a program parameter and commas inside numbers such as "10,000" do not trigger this feature.

4.4 Null Codes and Stop Words

The null code "---" is used to eliminate phrases that would otherwise be confused with actors or verbs. For example, the phrase "`Israeli-occupied West Bank and Gaza`" will generate both the `ISR` and `PAL` codes as actors. By adding the null code

```
ISRAELI-OCCUPIED [---]
```

only PAL is generated as an actor. Null codes have proven surprisingly important in refining the coding of the system, particularly in eliminating words that have the same stem as common verbs ("`acknowledgement`" versus "`acknowledge`") and eliminating verb phrases that do not correspond to political activity.

4.5 Issues

KEDS can code up to 9 sets of "issues": these are typically sets of words or phrases identifying the context or domain of an event. TABARI does not currently have this capability, but it will be added in the near future.

For example, one of the variables in the PANDA data set shows the topic being addressed by a political action such as a demonstration. This list begins with the phrases:

```
ABORTION [T]
AIDS_ [E]
ANCESTRAL LAND [N?]
APARTHEID [H!]
ASYLUM [H]
BALLOT [G?]
BAN_THE_DEATH_PENALTY [P?]
BANKING [F!]
```

Issue phrases can be coded as dominant (!) or subordinate (?), or given a numerical priority. The code for the issue with the highest priority will be assigned to the event.

4.6 Discard and Complex Events

Some news reports involve multiple international actors but no political events. Sports events are especially problematic given the propensity of Reuters to use national identities and martial metaphors in describing the athletic contests, particularly soccer ("`Algeria blasts Spain in World Cup action`"). In developing our Middle East data set, U.S. basketball star Michael Jordan was also a potential source of incorrect codes. Traffic accidents and natural disasters involving multinational fatalities are also a problem, as is transnational criminal activity (unless such activity is being explicitly coded).

Such stories are discarded by assigning key words and phrases the discard code ###

```
HEROIN [###]
MARIJUANA [###]
SOCCER [###]
WORLD_CUP [###]
```

If a discard phrase is found anywhere in the source text, no events are coded from the text. The question of what activities to discard and what to code depends on the research questions that the data are being analyzed. In the Middle East, activities involving illegal drugs, while certainly present, have relatively little political content. In contrast, when we coded a data set on Colombian and Mexican relations with the United States, illegal drugs were one of the paramount political issues. In those data sets, reports about illegal drugs were not only coded, but a specialized set of agent codes was developed to identify the various entities involved in the drug trade.

As noted above KEDS and TABARI can also detect a number of conditions where a sentence is likely to be too complex to code. For example, the sentence

> *Syria* said today the *U.S.* veto of a *U.N. Security Council* motion on *Israeli* settlements was "the most prominent phenomenon of *U.S.* hostility to the *Arabs* and *U.S.* support for *Israeli* plans to annex the *West Bank*"

contains nine actor references to six distinct actors (Syria, U.S., U.N. Security Council, Israel, Arabs, and Palestinians). The actors occur because a complex diplomatic process is being described—for example the object of the Syrian statement (“`U.S. veto of a U.N. Security Council motion on Israeli settlements`”) involves three actors. Unless the multiple actors are neatly arranged in compound phrases, TABARI usually fails to correctly sort out the subject and object from the modifying phrases. (In this instance, the program would also have incorrectly parsed “`Arabs and U.S.`” as a compound noun, rather than parts of two different compound phrases, leading it to erroneously code `ARABS <SUPPORT> ISRAEL`, assuming it was trying to code the verb “`support`”.) The simplest way to avoid such errors is to not code sentences that contain an excessive number of actors. Sentences of this level of complexity are not uncommon in Reuters, but in many instances the core event will also be reported in an additional, simpler sentence and therefore be coded.

The opposite problem occurs when a sentence contains *insufficient* actors. When coding lead sentences, the absence of an actor before the verb frequently indicates that the story does not involve a political event:

```
Kicking up dust, buffalo canter through low brush at this wetland oasis in
Jordan's eastern desert, once a world-famous sanctuary for migrant birds.
```

A sentence can also contain an excessive number of *verbs*:

The PLO, *raising the stakes* before *renewed* Middle East peace *talks*, has *accused* the U.S. of *cheating* Palestinians by *reneging on promises* to *grant* Israel $10-billion in *loan* guarantees only if it *halted* all *settlements* in occupied territories.

This admittedly unusual—but authentic—sentence contains seven verb phrases: "`raising the stakes`", "`renewed…talks`", "`accused`", "`cheating`", "`reneging on promises`", "`grant…loan`", "`halt…settlements`". Furthermore, several of the words found in the direct objects—"`talks`", "`promises`," and "`loans`"—could also be verbs, and are not preceded by determiners. If the comma-delimited phrase "`raising the stakes...talks`" is removed, TABARI will actually code the sentence correctly because the initial part of the sentence has the SVO structure "`PLO accused U.S`." But more commonly, multiple verbs are likely to cause coding errors, in part because many of the "verbs" are in fact nouns, or even adjectives (as in "`loan guarantees`").

A final problem that might cause a sentence to be too complex to code are intrinsically-ambiguous words. For example, "`Georgia`" can refer either to a region prone to armed ethnic conflict located in the southern part of the former Soviet Union, or a region prone to armed ethnic conflict located in the southern part of the United States. The headline "`Bomb blast in Georgia kills two people`" does not distinguish the two cases, although in most instances a lead sentence would contain sufficient additional geographical information ("`Tblisi`" versus "`Atlanta`") that the location would be clear.

5 TABARI FROM THE PERSPECTIVE OF PROGRAMMING

The discussion thus far has focused on TABARI as it would be viewed from the perspective of the user. This section will take a different perspective, that of the programmer. While the ultimate reference for TABARI is the source code itself, this is over 100 pages in length (including some internal documentation), and consequently a "top-down" overview may be useful.

Design Considerations for KEDS/TABARI

I will start this section with the admission that the design considerations below were not written down in 1988 and faithfully adhered to thereafter, but rather they evolved with the project itself. The hallmark of the KEDS program has been a pragmatic approach—we needed the data, and in the early stages of the project, there was considerable skepticism as to whether automated coding was possible.[1] The program was thus developed rather cautiously.

Simplicity

By the standards of natural language processing, TABARI, like KEDS, is a relatively simple program, with most of the processing done using a small set rules. The program does not attempt to deal systematically with complex features of language such as conditionals, tense, or recursive structures, although some of these can be dealt with idiosyncratically with verb patterns. Program development has generally been inductive rather than theoretical—techniques that work get incorporated into the program; those that do not (or which are too complicated for coders to effectively utilize.) are dropped.[2] TABARI does, however, contain a more complex language model than that found in systems that are based only on Boolean and proximity relationships between keywords, such as the programs used in most search engines and most content analysis programs (Alexa and Zuell 1999).

Robustness

One of the key constraints on the complexity of TABARI has been the erratic nature of the source text. Not only does the domain of political activity involve a far greater vocabulary and set of activities than, for example, the interactions involved in ordering food in a restaurant, but the text input is unedited except for the reformatting done to get sentences into the proper input format. It is, in fact, entirely unpredictable: some downloads contain incorrectly transmitted

[1] The motivation for KEDS corresponded closely to the sentiment expressed by Paul Baran, the RAND Corporation scientist who developed the “packet switching” concept that forms of the basis of information transfer in the Internet: “”I’ve learned that it is a hell of a lot easier just to build something than to try to convince somebody who doesn’t believe it’s possible.” (*Wired* March, 2001, pg. 151)

[2] For example, KEDS contains a fairly elaborate system for setting up interpreted transformational rules that can modify the grammatical structure of a sentence. However, as best I can tell no one has really used this, and while it checks out against the test cases I used when programming, there are probably still some bugs in it. I’m not sure whether I will try to implement this feature in TABARI, since those rules could be more efficiently implemented as compiled open-source code.

characters; some sentences are syntactically incorrect. No assumptions can be made about the relationships between words; there will always be an exception to any rule.[1]

While the robustness requirement was imposed out of necessity, it interacts synergistically with the simplicity criterion. Rather than trying to deal with all possible contingencies, the program instead simply gives up on sentences that don't correspond to a fairly small set of grammatical structures. This appears to have few adverse effects on the resulting event data, however, since news reports tend to be highly redundant.

Because of the robustness requirement, dictionary development has generally been cumulative because the underlying language model has not changed significantly since the early phrases of the project. We've followed a few dead-ends, and some unnecessary phrases were introduced into the dictionaries in order to get around errors in earlier versions of KEDS, but much of the coding is still done using phrases that were added to the dictionaries a decade ago.

Speed

TABARI is quite fast, processing thousands of sentences per second. This is partly a consequence of the simplicity of the sparse-parsing approach, but presumably has something to do with the internal structure of the program as well. The system generally works with very simple data structures, and while it is written in C++, it uses virtually none of the computationally-intensive elements of object-oriented programming.

Is this speed necessary? An event data coding system often will be processing hundreds of megabytes of text, so even small increments in the speed of processing each sentence can translate into hours of difference in the time required for re-coding. On the other hand, the speed of the underlying hardware continues to increase exponentially according to Moore's Law, and TABARI is running on hardware a least 200-times faster than the hardware originally used for KEDS. Coding a data set over a lunch hour, or even overnight, is hardly working at a snail's

[1] When I was developing TABARI, I used a test suite of about 100 sentences that contained all of the linguistic features that the program was supposed to be able to interpret correctly. Proper coding of test suite insured that new features did not have side-effects that caused existing features to malfunction, a frequent complication. Once the program passed this initial test, I also ran it on a set of about 26,000 Reuters lead sentences from the Middle East.

The Macintosh version of the program was compiled using the Metrowerks *CodeWarrior* compiler; the Linux version on the GNU *gcc* compiler. *One sentence* out of the 26,000 Reuters leads caused the Linux version of the program to crash even though the code had passed that test on the Macintosh. The problematic sentence involved a very strange—though grammatically correct—construction that caused a fault due to differences in how the two compilers did initializations.

pace. On contemporary computers, speed is also irrelevant from the perspective of a coder doing dictionary development because the time required to process a single sentence is almost instantaneous.

Yet something keeps telling me that there should be a niche for a very fast parser-coder. For starters, if one is already making compromises with sparse parsing, the least one can expect in return are the advantages of a fast implementation of that parse. Furthermore I still have the dream of real-time coding of large data sets where the coder could make a change in a dictionary and watch how that change affected, say, a 20-year time series. We are, in fact, quite close to having such a capability.

Will that make a difference in event data analysis? Possibly. Consider the analogous change that occurred in the practice of statistical analysis when, instead of submitting a box of punch cards at the computer center, then picking up the output the next morning, one can specify a regression model, estimate it, and plot the residuals in color in a couple of seconds. Or the evolution of word processing from embedded commands that were only implemented when the document was printed, to “what you see is what you get” dynamic formatting on the screen, to the real-time operation of spelling and grammar checkers. None of these changes implemented new features, but the fact that they could be done in real time changed the way people worked.

Multi-platform

KEDS was implemented only on the Macintosh operating system. This was done as a matter of personal choice: I have primarily worked with Apple equipment, find Microsoft systems to be awkward and designed for an office environment that is entirely different from that of research computing[1], and originally intended KEDS as a experimental system that would be employed only at Kansas.

Despite this intention, KEDS gradually gained a wider following—most notably through its early adoption by the PANDA project at Harvard—and it became clear that the Macintosh-only environment was inhibiting further use. I made several half-hearted attempts to port the system

[1] Ah heck, let’s be more honest: like most programmers who have not been brought up in a Microsoft-only environment, I passionately dislike the company, its products, its monopolistic position, and its tendency to quash innovation. However, I will give Bill and Melinda Gates credit for their recent philanthropic activities towards efforts to cure diseases affecting the Third World, a decidedly more admirable use of wealth than the habit of some high tech zillionaires of buying corporate jets for their teen-age sons.

to Windows, but it quickly became clear that because KEDS had been written without any considerations about separating the operating-system-dependent features of the interface from the data processing, this would effectively involve maintaining two separate programs.[1] This did not seem practical.

(The one concession that KEDS made to portability was keeping almost all of the input and output files as ASCII text files. This allowed individuals who were in a predominantly Windows environment to do most of their pre- and post-processing of text and events in Windows, and use a Macintosh only for the coding. The PANDA project employed this technique, as have some graduate students. TABARI is extending this by using XML to code the components of input files. At present XML is only found in the "project" file—the one file in KEDS that is stored using an internal format rather than as text—but it will eventually extended to the dictionaries as well.)

TABARI, in contrast, was explicitly constructed to support multiple operating systems, and the code for user interface is isolated from the code that does the parsing and coding.. I originally developed the program the program on a Macintosh, then ported it to Linux. About a month after the open source code was released, Dale Thomas, then a Ph.D. student at South Carolina, produced a Windows version, and a Solaris Unix version is in use by the U.S. government.

At the present time, the price of this flexibility is that TABARI has only a simple, text-only "teletype" interface. KEDS, in contrast, makes use of the standard elements of a "graphic user interface" (GUI). The GUI interface is not only more familiar to almost all computer users, but it can also be used to support forms of display that facilitate coding. For example, when I am working on dictionary development, I almost always work in the "parsed display" mode that color-codes the various parts of speech (actors, verbs, conjunctions, pronouns, and so forth) and crosses-out the words that are not being considered in the coding (generally subordinant clauses).

In principle, it should be possible to support GUI interfaces across multiple operating systems. The Stata statistics system, for example, is released simultaneously in Windows,

[1] The choice of Pascal also become an obstacle to porting the system. Unlike the situation with contemporary ANSI C/C++, critical differences existed between various Pascal dialects, in particular between Symantec's "Think Pascal" compiler used to develop KEDS and the Pascal compilers available for Windows. Because some of these differences involved string processing, there appeared to be a good chance that not only would I need to maintain separate versions of the code, but the Windows version would probably require substantial additional debugging.

Macintosh, and Unix versions, and the operation of the program is almost identical between Window and Macintosh. Given that the KEDS interface is not particularly complicated, it should be possible implement this in the near future.

Extensible, open-source code

Finally, where KEDS was the often idiosyncratic product designed for and by a single programmer, TABARI is *designed* to be extended. This is quite different from saying that it *can* be extended—Thomas's work on the Windows interface is the only additional work that I am aware of—but that code is open-source, the various functions (for example lexical analysis, syntactical analysis and coding) are compartmentalized, and the program is internally-documented. I've read—if not always faithfully followed—McConnell (1992) and Maguire (1993), and while TABARI has its share of idiomatic C code, it doesn't use any methods beyond those that could be mastered in a two-semester college-level computer programming class.[1] In theory, it should be possible to add new features to TABARI with only minor changes to—and without damage to the existing functions of—the existing source code.

In addition to all of the standard arguments for open-source code, I think it could be particularly important in a task such as automated event coding where the optimal feature set is still unknown. The sparse-parsing approach, which adds features incrementally as they are needed, rather than implementing an over-all plan, also puts a premium on the possibility of open-source enhancements of the code.

So, post the code and programmers will come? Not yet, at least. Notwithstanding a few exceptions such as Scott Bennett, Gary King, Charles Taber, and Michael Young, computer programmers are substantially rarer in the political science community than, say, statisticians with a comparable level of expertise. Event data coding remains a very specialized activity unlikely to draw a community of supporters comparable to that supporting Linux or Perl. While most political scientists who support event will probably work on dictionary development and the creation of new data sets—and the professional rewards for those activities may be higher—contributing to the code remains a possibility in an open source environment.

[1] I should, however, caveat this with the observation made by composer George Shearing about his 1952 jazz classic *Lullaby of Birdland*: "Took me ten minutes to write that song. Ten minutes and 25 years in the business."

Some Random Comments on the Code Itself

If one is contemplating working on the code, or perhaps taking some of the features of TABARI and implementing them elsewhere, the following comments may prove useful. My choice of topics here has been guided by the most common questions that I have been asked over the years about the programming of KEDS and TABARI. As always, the definitive reference on this topic is the source code.

First, while TABARI is compiled as an ANSI C++ program, it is in spirit a C program written with a bit of C++ syntax (likewise, KEDS was essentially a C program written in Pascal.). A few characteristics of C++ classes are used, as are some C++ I/O functions, but one will find little use of sophisticated object-oriented features of C++ such as inheritance, and nothing of the C++ STL. At the present time, TABARI does not even use the C++ <string> class. All objects are created at the initialization of the program, then kept intact, so the ability of C++ to effectively create and destroy complex objects is not used.

The failure to use the rich set of functions found in the C++ <string> class might seem surprising for a natural language processing program, but except in its input-output functions, TABARI moves away from string representations of the text very quickly. Character units are stored in a large block of contiguous memory; they are accessed through pointers to the beginning of the string, which is then null-terminated.[1] Sentences and verb phrases are broken down into unique “literals” that are represented internally as integers. Every distinct word or sub-word unit (for example a suffix) found in the dictionaries is assigned a unique integer. When a sentence or phrase is read into the system, it is decomposed into a list of tokens and connectors—for example, are two literals concatenated, adjacent (separated by only a space), or separated by intervening words?—and subsequent analysis is done using only the literals. This means that a 200-character sentences will typically be reduced to 20 or few integers, with a corresponding increase in processing speed.[2]

[1] This storage scheme allows C <string.h> library functions to operate on the stored characters, but a C++ <string> class function could not. KEDS used a similar storage system. That said, I’ve seen a number of benchmarks indicating that with contemporary compilers, operations using the C++ <string> class are as fast as equivalent operations using <string.h>, and I may shift some of the input and output operations to C++. Dictionary initialization was painfully slow in KEDS, but this is not a noticeable problem in TABARI.

[2] The use of literals is one of the main differences between KEDS and TABARI. KEDS kept the entire sentence throughout most of its processing, whereas TABARI throws away everything that it would not be able to recognize in the future; that is, anything that is not in a dictionary. This also means that verb patterns can be

Table 1 gives a brief synopsis of data structures used in TABARI; the names correspond to the variable names in the program itself. While these do not form a strict hierarchy—for example the storage of codes is rather idiosyncratic—they generally move from character strings, to literals, to lists of literals and connectors, to lists of those lists. In terms of fundamental data structures, the system gets no more complex than a one-way linked-list of records, though that structure is used extensively and some of the records themselves are links to other lists.[1]

The data structures and procedures used in coding, in contrast, are differentiated primarily by task rather than forming a hierarchy. Each level of processing—textual, lexical, syntactical and coding—does a series of operations on the data structure given it by the previous level, and then hands-off a new structure to the next level (see Tables 1 and 2). The various parsing and coding tasks outlined earlier in the paper are generally implemented as distinct functions that operate sequentially. In other words, the parsing process as described in the manual is generally the coding process that is implemented in the source code, and an experienced coder usually has a mental model of the coding process that corresponds closely to the actual algorithm.[2].

As much as possible, the individual parsing and coding functions are kept separate. This compartmentalization of functions was based on my experience with making modifications to KEDS. Some of these went very smoothly, and important new features could be added with minimal effort and no unintended side-effects. But parts of KEDS were a mess of inter-woven functions that should have been kept separate—this was particularly true of the handling of

---

stored as literals, which increases the speed of the recognition process. KEDS, in contrast, stored patterns as characters, and did a character-by-character comparison to identify matches.

[1] Dictionary look-up is handled with an array keyed on the first two letters of the string followed by a linear search—basically a very simple hash function—rather than a more sophisticated structure such as a tree. As usual, my choice here is a combination of pragmatism and a desire for simplicity. The number of items stored in the dictionary is unlikely to exceed 10,000, and with 676 (= 26^2) entry points in the array, an average of about 14 items would be found at each entry, so on average only 7 comparisons would need to be made to locate a word. A binary tree, in contrast, would require an average of about 13 comparisons.

In fact, these are only very rough approximations—initial letter ordering is distributed very asymmetrically in English, as are the words that must be looked up, and the number of literals is currently closer to 5,000. Nonetheless the simpler structure works well enough, it is simple to program and debug, and until such point that the vocabulary becomes so large that literal assignment significantly slows the program, the use of a more sophisticated structure does not appear justified. Even the exact relationship between the size of the vocabulary and the look-up speed is unclear, since the most time-consuming look-ups occur with words that are *not* in the dictionary at all but would otherwise occur at the end of the existing list.

[2] KEDS went through a period of intense debugging when the PANDA project first began to use the program for purposes that had never been tried at Kansas. One of their coders, Brad Bennett, had no training in computer programming, but could explain mistakes that KEDS was making in such systematic detail that sometimes I would know within a couple of lines of code where the bug was located.

compounds, and the failure to separate syntactical markup from coding. There are undoubtedly similar mistakes to be found in the design of TABARI, but it is at least the second iteration.

At the syntactical level, the list of literals is marked with tags that are similar to the XML-like tags that are displayed on the program screen, and TABARI is designed so that eventually text could be stored in marked-up form, and coding done independently. This would allow a sophisticated (and possibly quite slow) parser to mark-up the text off-line, while subsequent coding could be done in real time. Under the current scheme, dictionary modification would be limited to using vocabulary that was already in the system. Because much of our current dictionary development involves adding new vocabulary, this wouldn’t be very practical. However, if the marked-up output were the product of a more sophisticated parser with a nearly-complete dictionary, it might be useful.

Table 1: Data Structures

Dictionaries	
charStore	stores all character strings
codeStore	stores actor and event codes with their contingencies (e.g. date restrictions)
literals	storage of primary text units (words and suffixes) and their associated integer indices
tokenStore	assorted lists and linked lists of literal indices—for example synonym sets and suffix sets
phrase	lists of literals and connectors (appended, adjacent, proximate, unlimited)
patterns	phrases plus associated codes and special operators
roots	phrases, word type (actor, verb, conjunction, etc), default code, and pattern list
Coding	
textStruct	original text modified by standardizing punctuation and marking upper case
lexStruct	lexical processing—individual words are associated with literals, then assigned a word type
syntStruct	syntactical processing—identification of compound phrases, compound phrases, pronoun references
eventStruct	construction of event records; elimination of duplicate records

Table 2: Major Procedures

Procedures related to dictionaries

Dictionary Input and Output

Separate procedures for handing phrases and codes, as well as code contingencies.

Dictionaries are stored in an internal format, then reconstructed for output

Dictionary Modification

Separate procedures for handing phrases, codes, and contingencies, but verb and actor modification are handled through similar procedures. This is currently done using a keyboard-driven “teletype” interface.

Procedures related to coding

Lexical markup

Locate literals

Determine word type

Syntactical markup

Evaluate complexity conditions

Delimit compound nouns

Eliminate comma-delimited subordinate phrases

Determine pronoun references

Delimit compound clauses

Forward subject references across compound clauses

Coding

Locate verb

Check all patterns for the verb

Find source and target associated with the verb

Resolve code hierarchies and date restrictions

Eliminate duplicate records and invalid records (e.g. events with same source and target)

6 EVALUATION OF THE SPARSE PARSING APPROACH

The KEDS/TABARI sparse-parsing approach is far simpler than many systems now available for natural language processing. The drawback to this simplicity is that it is easy to find sentences that a sparse-parser will incorrectly code. The advantage of simplicity is that KEDS and TABARI can be easily adapted to a variety of problems, and the data produced by automated coding is now routinely used in quantitative political analysis published in refereed journals.

Both KEDS and TABARI were designed to be used (and have been used) by political scientists who have a great deal of knowledge about politics, but relatively little experience with linguistic theory and computer science. The system is designed for and by political analysts, and has been successfully used in a number of places outside of the University of Kansas. While there is definitely a learning curve involved in determining how to create dictionaries, this only involves the basic knowledge of natural language that an individual would acquire in a secondary-school education. In addition, the KEDS/TABARI dictionaries have an intuitive structure that most researchers pick up very quickly.

That said, it is important to keep in mind that any data set coded using the sparse-parsing approach will contain a number of erroneously-coded records, and any analytical method that uses that data must be insensitive to such errors. In some cases, these errors occur because dictionaries are incomplete or improperly updated. For example, I recently had a phone call from a government analyst who had acquired an anonymous event data set on China that he determined had been coded using KEDS (he recognized this because we have added a few codes to the original WEIS system). This data set recorded regular missile attacks between China and the USA. In all likelihood these errors occurred because our Middle East dictionaries had been used for coding, and in the Middle East the word “missile” usually signals an actual attack, such as those across the border between Israel and Lebanon, during the Gulf War, and in the Iran-Iraq War. In contrast, the word “missile” in US-Chinese relations usually concerns disagreements over Chinese exports of missiles and missile technologies. Additional work on the dictionaries could have eliminated many of these errors.

But other errors occur because of the limitations of the sparse parsing approach. These cannot be corrected within the framework on sparse parsing, or else could be corrected, but the phrases involved would be so idiosyncratic that it is unlikely that they would be encountered again.

Consider the following Reuters lead sentence:

```
A surprise setback today hit efforts to end the war between Israeli forces
and Palestinian guerrillas when Syria rejected the idea of the entire
Palestine Liberation Organisation moving to its territory
```

This sentence places the primary codeable event—"`Syria rejected`…"—far from the beginning of the sentence, contains a subordinate clause that is not delimited by commas, and uses the verb "`hit`" metaphorically. In fact, this sentence is not even readily interpreted by this native speaker of English, though the meaning is clear after the second or third reading. Such a sentence structure is far beyond the capability of a sparse-parser.

There are, however, at least three arguments against developing a more sophisticated parser. First, sentences too complex to correctly code using the sparse-parsing approach probably account for fewer than 10% of all Reuters leads, and some of those sentences will still be assigned the correct code for the incorrect reason. There is also considerable redundancy in news reports: the lead sentence given above probably comes from a story focusing on a political analysis, and may have been accompanied by a story about the Syrian rejection that had a lead with a simpler SVO structure, so the underlying event would have been coded even if this sentence could not be coded. An additional 5% to 10% of the sentences are those where even human coders will disagree, either because of ambiguities in the report or, more commonly, ambiguities in event coding schemes such as WEIS.

Second, I am not aware of any "off-the-shelf" parsers that perform substantially better than 90% on unedited source text, though this could change in the future as the capacity of computers and the sophistication of natural language processing models continues to increase. At present, there are numerous parsers and natural language understanding systems that are capable of doing much more sophisticated coding and classification than KEDS and TABARI in *limited* behavioral domains and/or with *pre-edited* text, but that is not the problem we need to solve. Our programs are designed to take *unedited* sentences from Reuters and other news services covering a very *wide range* of political behavior. Because event data set are coded from tens of thousands of sentences, the speed of the parser is also important, and full-parsing is often substantially slower than sparse-parsing.

Finally, a more sophisticated approach will probably involve semantic information—information pertaining to word meanings rather than simply their type and relationship to each other—and this will involve a very substantial amount of work in terms of dictionary development. For problems more complex than event data coding, parsers with semantic information are often essential, but we suspect they are not needed for our problem.

This is not to say that an improved machine-coding system could not be created; in fact I would be absolutely delighted to see such an effort undertaken. It is simply to say that such a project will probably involve a substantial amount of work and from the standpoint of event data research, it is more important to concentrate on refining the event coding schemes and the methods of analyzing event data rather than squeezing out another 5% accuracy, particularly since that additional information may make very little difference in the statistical results obtained from the data.

5.3 Next Steps

There are several features that could improve the coding accuracy and flexibility of TABARI, and some of these may be implemented in the next year. Many of these are relatively straightforward but were never implemented in KEDS either because of some early design decisions made in KEDS that could not be corrected until TABARI was written, or because they were not required for simple event data analysis.

Automatic generation of lists of proper nouns

The *Actor_Filter* program mentioned earlier is a first step in this direction, but is not integrated into the coding system. It would be useful to both recognize proper nouns and determine the likely relationships of these nouns to existing proper noun categories. For example, if the system recognized "`Hobbit Liberation Front`" as a category `HLF` and encountered the unknown proper noun "`Frodo Baggins`" in the context, "`Frodo Baggins, a leader in the Hobbit Liberation Front`", it would tentatively assign "`Frodo Baggins`" to the category `HLF`.

Systematically deal with time and conditionals

Reuters stories frequently deal with events in the near future or the recent past; TABARI does very little to distinguish these. To the extent that one can depend on a news agency weighting of the importance of various events by the extent of its coverage of those events, this can actually be useful: An activity that a news agency deems important—for example, a meeting between chief executives or between antagonists—will generate many more events than an activity that is routine, such as a meeting between agricultural ministers of two allies. However, a cleaner event sequence would be generated by coding only events that occur in the present, and this would also eliminate some of the widely miscoded events that result from reports of historical events and anniversaries.

Discussion of conditional events that *might* happen also occur, though this is more of a problem in full-story coding than in lead-sentence coding. At the present time, for example, we do not attempt to code direct quotes occurring inside stories, because frequently these involve complex conditionals that could be mistaken for actual events.

Make extensive use of sets of synonyms

TABARI dictionaries are organized by actors and verb phrases, rather than by codes. While this makes sense from a *parsing* perspective, from a *coding* perspective, organizing by code might make more sense—in other words, each event code has a set of associated verbs and a set of permissible objects. For example, a subset of the code for "give financial aid" might look like

```
Verbs:   [promise, pledge, grant, allocate, award]
Objects:[dollars, DM, yen, ecus, financial_aid, financial_assistance,
         bailout, debt_relief]
```

The obvious advantage of this scheme is that it is equivalent to 40 verb phrases. A secondary advantage is that it might also result in more transparent and easily-modified coding schemes, which could use existing lists of equivalent verbs and objects. Coding dictionaries could be organized around these sets of equivalent words rather than around individual words. In some cases these synonyms will be based on natural language usage—the *WordNet* synonym sets (`http://www.cogsci.princeton.edu/~wn/`) are a particularly important resource here—and in other cases the sets will be specific to a type of behavior being coded.

Determining pronoun references across sentences

An impediment to coding full stories is the ability to handle pronoun references to earlier sentences

```
Islamic foreign ministers meeting in Burkina Faso have prepared a draft
resolution condemning Israel for its blitz of Lebanon last week and
demanding that it pay reparations, officials said.

On Iraq, they call for a lifting of sanctions in return for Iraqi respect
of U.N. Security Council resolutions.
```

Based on our work on the Middle East, we originally thought that cross-sentence referencing would be relatively easy to solve, since pronouns in those stories usually refer to the first actor in the previous sentence, as in this example. However, this seems to be a quirk of the Reuters editors handling the Levant, and this technique has not worked as well in some other regions of the world. Determining pronoun references *across* sentences is more difficult than determining the references *within* sentences because the previous sentence may contain a number of different actors. As noted earlier, at times the problem is simply unsolvable without using contextual information not found in the sentences themselves. Nonetheless there are a number of methods available in the natural language processing literature that could be applied to it.

Bayesian verification of event coding

In such a system, the probability of an event assignment would adjusted depending on the existing pattern of events between a pair of actors. The goal of Bayesian verification is to eliminate wildly-inaccurate coding such as the use of force between two allied actors. Such incorrect codings usually occur because of an unusually-structured sentence, a reference to an historical event, or an unrecognized verb. A Bayesian verification system would continually adjust the threshold required for a particular event to be accepted as valid. If a disparate code was noted, the system will require greater validation (for example in the form of multiple reports, or multiple indicators within a single story that the code is accurate) than if the code is typical for the dyad. The system would also keep track of certain key events (for example the outbreak of armed conflict; a political transition due to a death, coup or election; natural disasters) that are likely to change the probability of certain types of events. This would probably considerably reduce the outliers coded by the system; it might have relatively little impact on the average behavior.

REFERENCES

Alexa, M. and C. Zuell. 1999. *A Review of Software for Text Analysis*. Mannheim: Zentrum für Umfragen, Methoden und Analysen.

Andriole, S. J. and G. W. Hopple. 1984. "The Rise and Fall of Events Data: From Basic Research to Applied Use in the U.S. Department of Defense." *International Interactions* 11:293-309.

Azar, E. E. 1982. *The Codebook of the Conflict and Peace Data Bank (COPDAB).* College Park, MD: Center for International Development, University of Maryland.

Azar, E. E., and J. Ben-Dak. 1975. *Theory and Practice of Events Research.* New York: Gordon and Breach.

Bond, D., J. C. Jenkins, C. L. Taylor and K. Schock. 1997. "Mapping Mass Political Conflict and Civil Society: The Automated Development of Event Data." *Journal of Conflict Resolution* 41:553-579.

Burgess, P. M., and R. W. Lawton. 1972. *Indicators of International Behavior: An Assessment of Events Data Research.* Beverly Hills: Sage Publications.

Daly, J. Ayres, and S. J. Andriole. 1980. "The Use of Events/Interaction Research by the Intelligence Community." *Policy Sciences* 12:215-236.

Esty, . C., J. A. Goldstone, T. R. Gurr, P. Surko and A. N. Unger. 1995. *State Failure Task Force Report.* McLean, VA: Science Applications International Corporation.

Esty, . C., J. A. Goldstone, T. R. Gurr, B. Harff, M. Levy, G. D. Dabelko, P. Surko and A. N. Unger. 1998. *State Failure Task Force Report: Phase II Findings.* McLean, VA: Science Applications International Corporation.

Gerner, D. J., P. A. Schrodt, R. A. Francisco, and J. L. Weddle. 1994. "The Machine Coding of Events from Regional and International Sources." *International Studies Quarterly* 38:91-119.

Goldstein, J. S., and J. C. Pevehouse. 1997. "Reciprocity, Bullying and International Cooperation: A Time-Series Analysis of the Bosnia Conflict." *American Political Science Review* 91:515-530.

Huxtable, P. A. 1997. *Uncertainty and Foreign Policy-making: Conflict and Cooperation in West Africa.* Ph.D. dissertation, University of Kansas.

Huxtable, P. A. and J. C. Pevehouse. 1996. "Potential Validity Problems in Events Data Collection." *International Studies Notes* 21,2: 8-19.

Laurance, E. J. 1990. "Events Data and Policy Analysis." *Policy Sciences* 23:111-132.

Lehnert, W., and B. Sundheim. 1991. "A Performance Evaluation of Text Analysis." *AI Magazine* 12:81-94.

Maquire, S. 1993. *Writing Solid Code.* Redmond, WA: Microsoft Press.

Marlin-Bennett, R., and J. C. Roberts. 1993. "Using Events Data to Identify International Processes: A New Unit of Analysis for International Relations." *International Studies Notes* 18:1-8.

McClelland, C. A. 1967. "World-Event-Interaction-Survey: A Research Project on the Theory and Measurement of International Interaction and Transaction." mimeo, University of Southern California, March, 1967

McClelland, C. A. 1976. *World Event/Interaction Survey Codebook.* (ICPSR 5211). Ann Arbor: Inter-University Consortium for Political and Social Research.

McConnell, S. 1992. *Code Complete.* Redmond, WA: Microsoft Press.

Merritt, R. L., R. G. Muncaster, and D. A. Zinnes, eds. 1993. *Theory and Management of International Event Data: DDIR Phase II.* Ann Arbor: University of Michigan Press.

Munton, D. 1978. *Measuring International Behavior: Public Sources, Events and Validity.* Dalhousie University: Centre for Foreign Policy Studies.

Salton, G. 1989. *Automatic Text Processing.* Reading, Mass: Addison Wesley.

Schrodt, P. A. 1994. Statistical Characteristics of Events Data. *International Interactions* 20: 35-53.

Schrodt, P. A. and D. J. 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, P. A. and D. J. Gerner. 1997. "Empirical Indicators of Crisis Phase in the Middle East, 1982-1995." *Journal of Conflict Resolution* 41:529-552.

Schrodt, P. A. and D. J. Gerner. 1998. "An Event Data Set for the Arabian/Persian Gulf Region 1979-1997." International Studies Association, Minneapolis, March 1998.

Schrodt, P. A. and D. J. Gerner. 2000. "Cluster-Based Early Warning Indicators for Political Change in the Contemporary Levant." *American Political Science Review* 94,4.

Schrodt, P. A., S. G. Davis and J. L. Weddle. 1994. "Political Science: KEDS—A Program for the Machine Coding of Event Data." *Social Science Computer Review* 12:561-588.

Sigler, J. H., J. O. Field, and M. L. Adelman. 1972. *Applications of Events Data Analysis: Cases, Issues and Programs in International Interaction.* Beverly Hills: Sage.

Thomas, G. D. 1999. *The "Strange Attractiveness" of Protracted Social Conflict in Northern Ireland.* Ph.D. dissertation, University of South Carolina.

Weber, R. P.. 1990. *Basic Content Analysis,* 2d ed. Newbury Park, CA: Sage Publications.