Automated Production of High-Volume, Near-Real-Time Political Event Data *

Philip A. Schrodt
Department of Political Science
Pennsylvania State University
University Park, PA 16802
schrodt@psu.edu

Version 1.0 : 31 August 2010

*Paper prepared for delivery at the Annual Meeting of the American Political Science Association, Washington, 2 - 5 September 2010. This project was funded by contracts from the Defense Advanced Research Projects Agency under the Integrated Crisis Early Warning System (ICEWS) program (Prime Contract #FA8650-07-C-7749: Lockheed-Martin Advance Technology Laboratories) as well as grants from the National Science Foundation (SES-0096086, SES-0455158, SES-0527564, SES-1004414). The views, opinions, and/or findings contained in this paper are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense, nor those of Lockheed-Martin or the National Science Foundation. The paper has benefitted from extended discussions and experimentation within the ICEWS team and our research group at the University of Kansas; I would note in particular contributions from Steve Shellman, Hans Leonard, Brandon Stewart, Jennifer Lautenschlager, Andrew Shilliday, David Van Brackle, Will Lowe, Steve Purpura, Vladimir Petrov, Baris Kesgin and Matthias Heilke. The results and interpretations are solely the responsibility of the author. The open source software discussed in this paper, as well data sets generated in earlier NSF-funded research, can be downloaded from the web site: `http://eventdata.psu.edu/`

Abstract

This paper summarizes the current state-of-the-art for generating high-volume, near-real-time event data using automated coding methods, based on recent efforts for the DARPA Integrated Crisis Early Warning System (ICEWS) and NSF-funded research. The ICEWS work expanded by more than two orders of magnitude previous automated coding efforts, coding of about 26-million sentences generated from 8-million stories condensed from around 30 gigabytes of text. The actual coding took six minutes. The paper is largely a general "how-to" guide to the pragmatic challenges and solutions to various elements of the process of generating event data using automated techniques. It also discusses a number of ways that this could be augmented with existing open-source natural language processing software to generate a third-generation event data coding system.

1 Introduction

The modernization of Japanese industry from 1950-1990, which saw the country go from being a producer of low-cost, low-quality mass-produced items such as toys and clothing, to producing of some of the most advanced technology on the planet such as the equipment used to fabricate microprocessors, was driven in large part by the concept of "*kaizen*", an easily-translatable concept that means continuous, incremental improvement. Japanese engineers realized that while major breakthroughs could occur—and the Japanese were certainly responsible for many—these were irregular, unpredictable, and often as not the result of individual genius. *Incremental* improvements, on the other hand, could occur on a daily basis, anywhere on the factory floor, and yet, over time, thousands of tiny improvements had at least as great an impact as one or two big ones. The result was the reduction in the size of a device for playing music from that of a suitcase to that of a piece of chewing gum, the decimation of the U.S. auto industry—until it also adopted *kaizen* methodologies—and the complete transformation of the role of Japan in the international economic system in the space of two generations.[1]

Political event data have had a long presence in the quantitative study of international politics, dating back to the early efforts of Edward Azar's COPDAB (Azar 1980) and Charles McClelland's WEIS (McClelland 1976) as well as a variety of more specialized efforts such as Leng's BCOW (Leng 1987). By the late 1980s, the NSF-funded *Data Development in International Relations* project (Merrit, Muncaster and Zinnes 1993) had identified event data as the second most common form of data—behind the various Correlates of War data sets—used in quantitative studies (McGowan et al 1988). The 1990s saw the development of two practical automated event data coding systems, the NSF-funded KEDS (Gerner et al 1994, Schrodt and Gerner 1994) and the proprietary VRA-Reader (`http://vranet.com`; King and Lowe 2004) and in the 2000s, the development of two new event coding ontologies—CAMEO (Gerner, Schrodt and Yilmaz 2009) and IDEA (Bond et al 2003)—designed for implementation in automated coding systems.

While these efforts had built a substantial foundation for event data—for example by the mid-2000s, virtually all refereed articles in political science journal used machine-coded, rather than human-coded, event data—the overall investment in the technology remained relatively small. The KEDS[2] project focused on coding a small number of regions, mostly in the eastern Mediterranean. VRA did global coding—and made this data available through Harvard University (`http://gking.harvard.edu/events/`)—and had an assortment of contracts with various NGOs and government agencies but, as a proprietary system, little of the underlying technology fed back into the academic research environment.[3]

[1] Managing real estate and equities bubbles and their consequences... not so good...

[2] Kansas Event Data System, a term used to refer to both the overall project which existed at the University of Kansas from 1990 to 2009, and the original coding program, which was written in Pascal for the Macintosh operating system. That project has now moved to Penn State but, despite months of effort, we have yet to come up with a clever name for it that does not produce an acronym that sounds like an obscure skin disease. Suggestions are welcome. Meanwhile we are just calling it the "Penn State Event Data Project."

[3] VRA-READER is based on a full-parser, in contrast to the shallow-parsing of KEDS/TABARI, and thus involves quite different principles, apparently a sentence-frame model linked very closely to the IDEA ontology, but beyond that little has been published about the system, and even the documentation of the public

This situation changed dramatically with the DARPA-funded Integrated Conflict Early Warning System (ICEWS; O'Brien 2010 and additional papers on this APSA-2010 panel) which invested very substantial resources in event data development, including both TABARI—the open-source successor to KEDS —and VRA (the latter used as one of the sources to generate the ICEWS "events of interest"; O'Brien 2010: 91). The first phase of ICEWS used a variety of approaches under several prime contractors; the suite of models developed by the Lockheed Martin Advanced Technology Laboratory (LM-ATL) passed a set of pre-determined benchmarks for out-of-sample accuracy, precision and recall on five indicators of political activity, and is being further developed. The LM-ATL efforts have made extensive use of TABARI, and that will be the focus of this paper.[4]

The key difference between the ICEWS event data coding efforts and those of earlier NSF-funded efforts was the scale. As O'Brien—the ICEWS project director—notes,

> ...the ICEWS performers used input data from a variety of sources. Notably, they collected 6.5 million news stories about countries in the Pacific Command (PACOM) AOR [area of responsibility] for the period 1998-2006. This resulted in a dataset about two orders of magnitude greater than any other with which we are aware. These stories comprise 253 million lines of text and came from over 75 international sources (AP, UPI, and BBC Monitor) as well as regional sources (*India Today, Jakarta Post, Pakistan Newswire,* and *Saigon Times*).

In Phase II, as we have moved to near-real-time event data production, the scale of this coding effort has increased even further.

The purpose of this paper is to describe a number of incremental improvements and lessons-learned in the experience of the LM-ATL team in dealing with automated event data. As O'Brien (2010) describes in some detail, event data is by no means the only forecasting approach used in ICEWS, but it was one important approach, and was a key element in several of the LM-ATL models. This paper is a "how-to" exercise—albeit at a rather high level of generality in places—rather than a theoretical one, and the objective is to provide some guideposts for others who might be interested in undertaking similar efforts, whether as basic research or for applied policy purposes.

2 Processing Tasks

2.1 Text acquisition and formatting

The first step in generating event data is the acquisition of news reports to code. Following the lead of most event data projects, we relied primarily on the Lexis-Nexis (LN) data service; these stories were acquired under contract by Lockheed and provided to us, generally already in the format required of TABARI, though in the initial phases we were doing some of the formatting as well.

---

data set is rather scant.

[4] I will be focusing on the open-source side which we pursued at Kansas and Penn State; in addition LM-ATL and some of its subcontractors have several proprietary efforts underway that produce event data using automated methods. Some of these are based on TABARI; others use different technologies.

The two key differences between this project and most earlier event data projects was the sheer magnitude of the downloads, and the fact that we were using multiple sources. The eventual text corpus for 1997-2009—after initial filtering—involved about 30 Gb of text, which reduced to about 8-million stories.[5] Second, unlike most earlier projects that used a small number of sources—typically the international newswires Agence France Press, BBC, Associated Press and United Press International[6]—we used about 30 different regional sources.[7]

The first *major* challenge that we—and numerous other projects, including some other additional NSF work I've been involved with (Schrodt, Palmer and Hatipoglu 2008)—has been problems with LN. The fundamental problem is that neither the LN organization nor, apparently, the LN technology is capable of handling a request for a large amount of data: Identical search strings run within minutes of each other will sometimes return vastly different results; sources known to be in the data base will be inexplicably excluded, and multiple attempts were needed before it appeared that we actually had all of the stories matching the search string, or at least all that we were ever going to get. Furthermore, these problems vary over time—sometimes they will be so bad that LN is essentially un-useable for a period of time (presumably during system upgrades), then matters will be [relatively] okay for a period of time. In all likelihood, this is due to LN using a legacy system that was designed to do very narrow searches, rather than providing a large-scale data dump. Unfortunately, LN is essentially a monopoly and there are no alternatives[8] so this is a necessary obstacle to overcome, at least when long-time series data are being acquired.

The use of multiple sources provides a challenge in extracting the required information—the date, source and individual sentences—from the original download. Following the earlier work in the KEDS project, we were largely using source-specific filters, generally in `perl`. While LN is generally consistently formatted, the diverse set of sources—and the sheer size of the files—proved a challenge, particularly since the local sources are more likely to contain minor quirks that will throw off a filter.

As we had discovered in earlier projects, in many sources the task of sentence delineation is a major challenge, both due to the presence of abbreviations, the occasional formatting errors that will cause sentences or entire paragraphs to run together, and the presence of a very large amount of non-sentence material such as tables of sports scores, exchange rates and commodity prices, chronologies, news summaries, weather reports and other such material.[9] These are sufficiently widely varied that it is nearly impossible to eliminate all of this using rules on the story itself—though we did have about 30 or so simple rules based on the

[5]The count of "stories" has varied continually as we've updated the downloads, modified the filters and so forth, and so an exact count is both unavailable and irrelevant. But starts around around 8 to 9-million.

[6]Many projects, including KEDS and VRA, have also used Reuters, but this is not available from LN, and was prohibitively expensive to acquire at commercial rates from Factiva.

[7]We've actually identified about 75 distinct sources in the stories, presumably the result of quirks in the LN search engine. However, these additional sources generate only a small number of stories, and by far the bulk of the stories come from the sources we had deliberately identified.

[8]Reuters/Factiva does not have these search engine problems, presumably because it is working with a relatively new system. However, Factiva only provides Reuters and as noted above, is prohibitively expensive, even for DARPA-funded research.

[9]In principle, a suitably complex LN search term should exclude these; in practice one can't depend on this and frequently those terms appear to have little effect at all.

headline of the story—and instead one needs to use more general rules such as the length of the "sentence." Most genuine news sentences are around 150 to 300 characters in length, and anything below about 40 characters is almost certainly not a legitimate sentence. There are also a few patterns easily written as regular expression that will identify non-material: For example something of the form `\d+\-\d+` is almost always a sports score.

2.2 Filtering: Irrelevant Stories

Irrelevant stories have been the bane of the event data source texts from the beginning of our experience. For example, our now-30-year "Levant" data set basically just looks for stories containing the names or synonyms of the six actors we are tracking: Egypt, Israel, Jordan, Lebanon, the Palestinians, and Syria. However, our early downloads covered the peak of the career of basketball player Michael Jordan and we ended up with quite a number of basketball stories. These are relatively harmless and easily discarded by TABARI (or the LN filter when it is working... again, it is not reliable in our experience so we usually depend on our own post-filtering), but they do present problems when downloading—which we originally did across a phone modem—or when one is paying by the story, as Lockheed has negotiated.

However, other types of stories are much more problematic. The most important are chronologies and retrospectives, which describe political events that occurred in the sometimes distant past, yet the dateline of the story is in the present. A good example would be various World War II commemorations, which typically receive extensive coverage and could be miscoded as conflict behavior between the US, Germany and Japan.

Another longstanding problem are international sports competitions that use military metaphors. World Cup reports, for example, always use the simple national names—Netherlands versus Spain—and not infrequently use terms such as "battle," "fought," "stand-off" and the like. These can usually solved by discard phrases[10] involving every imaginable form of competition, sporting and others. But even this will fail when the sports context is implicit, such as a [hypothetical] report on 11 July 2010 that might begin, with little concern that it will be misinterpreted, as even the USA now is aware of the World Cup, "Fans eagerly await tonight's battle between the Netherlands and Spain." Furthermore the sheer volume of such stories—as much as a third of the stories in areas where little seems to be happening except sports[11]—decidedly increases download times and costs.

2.3 Filtering: Duplicates

The LN downloads contain a very large number of stories that are either literally duplicates, or else are effectively duplicates. These generally come from five sources

- Exact duplicates, where a local source simply reprints the contents of an international newswire story. This is what newswires are for, so this happens a lot;
- Multiple reports of the same event—for example a suicide bombing—as it develops; AFP does this frequently;

[10] A TABARI discard phrase causes the story to be skipped if the phrase occurs anywhere in the text.
[11] In our downloads, Australia.

- Stories repeated to correct minor errors such as incorrect dates or spelling;
- Lead sentences that occur in general news summaries—which may occur multiple times during a day—as well as in the story itself;
- Multiple independent reports of the event from different news sources: this was a major issue because of the large number of stories we were coding.

Duplicate detection is a very difficult problem, particularly when multiple sources are involved. We dealt with exact and near duplicates by simply seeing whether the first 48 characters of the story matched—this obviously will catch all duplicates and tends to catch minor duplicates such as corrections of spelling errors much of the time.[12] Cross-source duplicates are dealt with using the *One-A-Day* filter discussed below.

When used in a predictive mode, as we are doing with ICEWS, duplicates are not necessarily a bad thing, since they generally will amplify politically-relevant signals. In other words, if reporters or editors think that something is important, it is more likely to be repeated, both within sources and across sources, than something that is mundane.

However, when trying to measure changes of "ground-truth" behavior against a baseline over a long period time, duplicates are a serious problem, both across sources and within sources. Cross-source duplication has probably changed considerably over the past fifteen years due to local sources putting increasing amounts of material on the Web, and more generally the globalization of the news economy, so that events in once-obscure places are potentially of international interest.[13] In-source duplication can change due both to changes in the resources available to an organization—while not part of the ICEWS source set, Reuters went through something close to an organizational near-death experience during the period 1998-2002 and the frequency of its reporting dropped dramatically during that time—and policies on updating, corrections and the broadcasting of summaries.

2.4 Dictionary Development

By far the greatest—and still incomplete—challenge of scaling-up the KEDS/TABARI system has been in the area of actor dictionary development. The KEDS project had focused on a small number of geographical areas, primarily the Levant, with ten-year data sets on the Balkans and West Africa. We had done some experimental work under small government contracts to code individual countries in other areas of interest, in all parts of the world, for short—typically two-year—time periods, and graduate student research by Ömür Yilmaz and Baris Kesgin had produced very detailed dictionaries for Turkey, but that was it. ICEWS, in contrast, involves coding 29 states that encompass more than half the world's population.

[12] This will not, however, catching spelling corrections in the first 48 characters. In the Reuters-based filtering for the KEDS project, we did a count of the frequency of letters in the lead sentence, and identified a duplicate if the absolute distance between that vector for two stories, $\sum |x_i - y_i| > \eta$, where the threshold η was usually around 10. This catches spelling and date corrections, the most common source of duplicates in Reuters, but failed on AFP, which tends to expand the details in a sentence as more information becomes available.

[13] Notably to traders—carbon-based and silicon-based—in the financial sector, which drives much if not most of the international reporting. The likelihood of an event being reported is very much proportional to the possibility that someone can make or lose money on it.

The earlier KEDS data sets were initially developed by individuals—largely undergraduate honors students[14]—who went through sentences item by item and added new entries to the actor and verb dictionaries as they encountered incorrectly coded items. This was later supplemented by a relatively simple named-entity-recognition program called *ActorFilter* that would locate potential new names based on capitalization patterns, compare these to entries in the existing dictionaries, and then produce a keyword-in-context listing of entities which appeared to be new, listed in reverse order of frequency. This was particularly useful in making sure that any major new actors were not missed, and was our first step in developing dictionaries for new countries.

Neither of these techniques scaled, particularly in the relatively short time frame of the first phase of the ICEWS work. While we did some spot-checking of individual stories, our ability to do this with any meaningful proportion of the 26-million sentences in the ICEWS corpus was limited. *ActorFilter*, unfortunately, had not been designed for a project of this magnitude and while it could be used on a sample, it slowed to an unusable crawl on very large files.

Fortunately, the *.verbs* dictionaries—which involved about 15,000 phrases—needed relatively little work to produce useable (if hardly perfect) data: this was consistent with our earlier experiments in extending the data sets, which have always used a shared *.verbs* dictionary despite using specialized *.actors* dictionaries. We did one experiment where we looked at a sample of sentences where TABARI had *not* identified a verb phrase, and this produced a few new candidate phrases, but relatively few. We did considerable work on cleaning up those dictionaries from the accumulated idiosyncrasies of two decades of different coders, but they remained largely unchanged.[15]

This would not be true for the *.actors* dictionaries, which occupied the bulk of our work.[16] Two approaches were used.

First, rather than deriving the actors from the texts, we tried to locate lists of actors and incorporate these into both international and nation-specific dictionaries. At Kansas, for example, we converted the information in the *CIA World Factbook* to a comprehensive list of state names, major cities, adjectival forms, and date-delimited lists of heads of state; we are currently in the process of extending this to include all government leaders from the *CIA World Factbook* monthly "World Leaders" listings, which are available as PDF files for 2001 to the present. Various national sources provided lists of parliamentarians and other local leaders, and we've also been expanding the list of NGOs and IGOs. As a consequence, the actors dictionaries now have around 20,000 entries, compared to the 1,000 or so entries typical in earlier KEDS work.

Second, we improved the ability of TABARI to automatically assemble codes from combinations of a named actor and an generic agent: how example "Philippine soldiers" will

[14]One of whom is now a full professor at the University of Wisconsin, Jon Pevehouse. We've been doing this stuff for a while...

[15]The same was true of the CAMEO coding ontology, despite CAMEO originally being intended specifically to code events dealing with international mediation (Schrodt and Gerner 2004, Schrodt, Gerner and Yilmaz 2009). This is probably due to the fact that while CAMEO was originally going to involve a minor, six-month revision of WEIS for a single NSF grant, we ended up spending almost three years on the project, with several complete reviews of the dictionaries, and hence effectively created a more comprehensive ontology.

[16]This was done both by individuals at Kansas and by Steve Shellman's researchers.

automatically generate the code PHLMIL, whereas "The Philippine Secretary of Agriculture" will automatically generate the code PHLGOV. We had been coding this type of substate information for quite a few years, but did so with separate dictionary entries for, say, "Australian police," "Cambodian police," "Chinese police" and so forth. The new system is both faster in terms of the dictionary size and much more efficient. This allows the coding of both generic agents such as "police", "soldiers", "demonstrators" and the like,[17] as well as named individuals where we have the title in the dictionary but not the individual person. For most of our coding, at least for the forecasting efforts in ICEWS, individual identities are not used, so this gets quite a bit of information we were previously missing. In support of this new facility, we also increased the size of the *.agents* dictionary considerably.

These efforts were a major step forward, but dictionary development—and maintenance, as dictionaries need to be updated as political figures change—remains a considerable challenge, which will be addressed below.

2.5 Cluster Processing

TABARI is an open-source C++ program—compiled under gcc—that runs on a common code base in both the Macintosh OS-X and various Linux/Unix environments, which proved useful in deploying it across a combination of desktop, server and cluster environments.

The major innovation in the 2009 coding was the use of a computer cluster to dramatically increase the coding speed. In the 2008 data development for ICEWS Phase I, coding the 1997-2004 data on personal computers required almost a week: this was also hampered by the existence of some bugs in TABARI that occurred only with very rare sentence structures and which thus had gone undetected in earlier work with the program, but emerged during that coding; there were initially eight of those out of the 26-million sentences.

In 2009, we gained access to a small, 14-processor cluster computer that was sitting unused (and undocumented) at the University of Kansas. Rather than trying to get TABARI to run in parallel at the micro level, we did "parallelism on the cheap" and simply split the text files to be coded across the processors, which share a common file space, and ran them simultaneously, then re-combined the output files at the end of the run. TABARI ran on the individual nodes at around 5,000 sentences per second; the throughput for the cluster as a whole ended up around 70,000 stories per second, allowing the entire 26-million story corpus to be coded in about six minutes. The initial set-up, of course, took quite a bit longer, but this was particularly useful for weeding out aforementioned problematic records that would cause the program to crash.

A 14-processor cluster is, of course, tiny—Penn State has multiple clusters available to social scientists that are in the 256-processor range—so effectively the coding speed is unlimited, even for a very large corpus. Furthermore, this can be done by simple file splitting, so the gain is almost linear.

[17] A still missing component of the system is the ability to tag the entire story with the location, which will allow the agents to be coded even if they are not preceded by a national identifier. This is particularly important in local sources: unlike an international news report, a Philippine news report on Mindanao, for example, will almost never mention that Mindanao is part of the Philippines. There are several software systems for doing this type of tagging and LM-ATL is experimenting with them; the required modifications for TABARI to use that information would be minor.

That said, we did not use the cluster for the other processing, since `perl` ran quite quickly on the Macintosh XServe hardware we were using to store and manage the text files, and most of the remaining steps only took a few minutes. The downside to working on the cluster is the time required to transfer the multiple gigabytes of text, and running `perl` on the server proved sufficient.

2.6 Post-coding processing

Following the protocols used in most of the research in the KEDS project, the major post-processing step is the application of a "one-a-day" filter, which eliminates any records that have exactly the same combination of date, source, target and event codes: this is designed to eliminate duplicate reports of events that were not caught by earlier duplicate filters. In our work on the Levant data set, this fairly consistently removes about 20% of the events and the effect on the ICEWS data is comparable despite the use of a greater number of sources.

In areas of intense conflict—where multiple attacks could occur within a single dyad in a single day—this could eliminate some actual events. However, these instances are rare, and periods of intense conflict are usually obvious from the occurrence of frequent attacks across a month (our typical period of aggregation), and do not require precise measures within a single day. Periods of intense conflict are also likely to be apparent through a variety of measures—for example comments, meetings with allies, offers of aid or mediation—and not exclusively through the attacks themselves. As a consequence—consistent with earlier work on the KEDS project—the one-a-day filter improves the quality of the data.

From this point, a variety of different things are done with the data, but these fall into the category of data management and model construction, rather than data generation per se. LM-ATL is developing an increasingly elaborate system for the management of the data that includes a wide variety of visualization tools, as well as interactive "drill-down" capability that allow a user to go from the coded events back to the original text, as well as management and display of the coding dictionaries. On the modeling side, the data can be aggregated in a variety of ways, including event counts for various types of dyads as well interval-level scaled data using a modification of the Goldstein (1992) scale for the CAMEO event ontology.

3 Tabari High Volume Processing Suite

The various operations noted above have been combined into an open-source "high-volume processing suite." As typical with a Unix data-processing approach, this was done with a number of small programs that were used to solve a specific problem, rather than general purpose utilities. These programs would almost certainly need some modification in order to be used on subsequent projects. Nonetheless, they worked, and might be easier than starting anew.

The programs are generally quite short. Remarkably, given the amount of processing involved, they were also quite fast—on a Macintosh XServe with two 2.3 Ghz quad-core

Intel Xeon processors and 3 Gb memory, all ran in less than five minutes.[18]

The programs are presented with brief descriptions in the order they were run. As much of the objective of this paper is simply to provide pragmatic guidelines for how to do this, the discussion below assumes that the reader is familiar with the TABARI program, or rather will give this paper to a graduate student who will read the manual (yes, RTFM!) and figure out what this means.

3.1 LN.data.select.pl

This takes a TABARI -formatted input file with standardized record headers and selects only those records with sentence numbers 001-004 (first four sentences of the story)—the coding norm we were using in this project—and where the first character in the story is not " or ' (direct quotes). It also breaks the file into 1,000,000 story blocks.

3.2 LN.data.sort.pl

This takes a series of input files and for each file, it moves the individual records into into a single string, then sorts on all of those strings. The TABARI format has the date as the first element in a record, and so this results in a chronological sort.

3.3 LN.data.filemerge.pl

This takes a series of *pairs* of sorted input files and merges them—it is intended to be run iteratively so to produce a completely sorted set of records.

This program is a bit awkward since it just works with pairs of files and, consequently, needs to be run $log_2(N)$ times. It reads directory lists from a dictionary, with modifications of the command line parameter, so a binary combination of involves a sequence of commands of the form

```
perl LN.data.filemerge.pl 1
ls merge.1* > sort.2.list
perl LN.data.filemerge.pl 2
ls merge.2* > sort.3.list
perl LN.data.filemerge.pl 3
ls merge.3* > sort.4.list
perl LN.data.filemerge.pl 4
ls merge.4* > sort.5.list
perl LN.data.filemerge.pl 5
ls merge.5* > sort.6.list
perl LN.data.filemerge.pl 6
```

Only trick here was that the current version requires a file list that has an even number of files, so in any step where that condition was violated I pulled out a file and inserted it later (in the set I was working with, this only needed to be done once). A minor modification of the program could deal with an odd-numbered file list automatically by just copying the

[18]Much of this work—particularly the sorting and merging—could also be done in a database environment such as mySQL. However, in the experiments we've done, the `perl` programs, precisely because they are highly specialized and do not have the overhead associated by a DB, are orders of magnitude faster.

final file, and with further modification, one could do the binary iteration in a loop, removing the need to run the program more than once. This additional code is left as an exercise.[19]

3.4 LN.data.dups.pl

This takes the final merged input file and removes duplicates within a single day based on the first 48 characters in the sentence. Unique entries are written to a new file and a file is generated containing a day-by-day listing of the unique and duplicate record counts.

3.5 LN.data.counts.pl

This little utility program takes a sorted input file and counts the number of records per day—it was used to do a consistency check on the earlier results.

3.6 LN.data.seqsort.pl

This takes an input file and restores it to record-sequence order following the running of `LN.data.dups.pl`.

3.7 LN.data.divide.pl

This takes an input file and copies only those records meeting a hard-coded date criterion—this is used to eliminate records that were either out of range due to quirky LN search results, or badly-formed dates, and generates the final chronological ordering of the data.

3.8 TABARI

This processing eventually produced about 26-million sentences. These were split into 26 files using `LN.data.divide.pl`, then uploaded onto a 14-node Linux parallel cluster. We created a series of distinct project files that contained the same dictionaries, but differed in the $< textfile >, < eventfile >$ *and* $< errorfile >$ fields, e.g.

```
<textfile>    /var/tmp/LN.date00.06.3
<textfile>    /var/tmp/LN.date00.06.4
<eventfile>   LN.4.evt
<errorfile>   ICEWS.4.errors
```

In the system we were using, it made sense to move the individual text files to the */var/tmp* directory, which is separate for each node. Your configuration may differ.[20] The various *.verbs*, *.actors*, and *.options* dictionaries, however, were read from directory common to all of the processors, so these were common to all of the runs.

Once this has been set up, just run

[19] It may also be entirely unnecessary: some later experiments have suggested that the `perl` memory management is sufficient to handle even multi-gigabyte files and this approach may be unduly conservative in the current environment.

[20] Your configuration may also have documentation... this work having been done at the University of Kansas, ours didn't...

```
./TABARI.0.7.2b2 -a LN.coding.project.n
```

in each node of the cluster, where n corresponds to the project number. The $-a$ command line option invokes the `A)utocode N)one` command which, in combination with a $<$ *coder* $>$ line in the *.project* file, means the program codes with no further intervention. It should be relatively easy to add formal parallel processing code to TABARI using the MPI system (which was not properly installed on the KU system), but this simple method is quite workable. For future runs at Penn State, where MPI is installed, we expect to do this.

3.9 HV.eventmerge.pl

This merges a set of event files produced by an ordered list of TABARI *.project* files, processes each to determine the most recent version, then just concatenates those files. It was written to combine the event files produced by multiple copies—and potentially, partial runs—of TABARI running on a cluster.

4 Real-Time Coding

In addition to the near-real-time coding system described above—data are currently updated once a month—we also used the 2009-2010 period to undertake an experiment in true real-time coding using RSS feed. This work was done largely by Vladimir Petrov, a KU Ph.D. student, and will be continued at UT/Dallas under NSF funding.[21]

RSS feeds present a potentially very rich source of real-time data because they are available in actual real time using standard software, and, of course, are free. The downside of RSS feeds is the absence—at least at the present time—of any archival capacity, so they can be used for current monitoring but not for generating a long time series.

A variety of RSS feeds are available. The richest would be two major RSS aggregators, GoogleNews and European Media Monitor, which track several thousand sources each. In some experimental downloads in 2008, we found that these were generated about 10 Gb of text per month, and that volume has probably only increased. The two downsides with the aggregators are massive levels of duplication, and the fact that they are not produced in a standard format: instead, each source must be reformatted separately. This is not particularly difficult in terms of simply detecting the natural language text of the news report itself—and in fact all of these feeds consist largely of HTML code, which typically takes up more than 90% of the characters in a downloaded file—but can be difficult in terms of detecting dates and sources.

Instead of looking at the aggregators, we focused on two high-density individual sources: Reuters and UPI. In addition to providing RSS feeds, these also have archives, back to 2007 for Reuters and back to 2001 for UPI; these could be downloaded from the Web. The focus on individual sources meant that only a small number of formats had to be accommodated—even formats within a single source exhibit some minor changes over time—but these two sources, as international news wires, still provide relatively complete coverage of major events. They

[21] The hardware was moved from KU to UTD over the summer but as of this writing is still not online with a public URL. However, we anticipate the transfer process to be completed in the very near future and the system will be available for public use.

do not, however, provide the same level of detail as the commercial sources, Factiva for Reuters and LN for UPI. After some experimentation, it turned out to be easier to access the updates to this information from their web sites rather than through RSS feeds per se, but this still allows fairly rapid updating.

Implementation of a real-time coder was a relatively straightforward task of linking together, on a server, the appropriate reformatting and duplicate detection programs, running TABARI at regular intervals on the output of those programs, and then storing the resulting event data in a form that could be used by other programs: mySQL was used for this purpose. Petrov developed a prototype of a user-friendly interface for basic data display, and we will be extending this in the future at UTD with the intention of eventually providing a readily-accessible source of global event data that can be used for real-time monitoring, analysis and forecasting.

While the basic implementation of this system has been relatively straightforward, our 18-month experiment has found at least three characteristics of the data that should be taken into account in the design of any future systems.

First, while *in principle* one could get real-time coding—automated news monitoring services used in support of automated financial trading systems routinely do this—there is little reason to do so for existing event data applications, which generally do not work on data that is less finely grained than a day. Furthermore, the news feeds received during the course of a day are considerably messier—for example with minor corrections and duplications—than those available at the end of a day. Consequently, after initial experiments we updated the data only once a day rather than as soon as the data became available.

Second, these are definitely not "build and forget" systems due to the changing organization of the source web sites. Reuters in particular has gone through three or four major reorganizations of their web site during the period we have been coding data from it, and in one instance was off-line for close to a week. Thus far, the changes in code resulting from these reorganizations have been relatively minor, primarily dealing with the locations of files rather than the file formats, but it has necessitated periodic—and unexpected—maintenance. The RSS feeds may have been more reliable—these presumably did not go off-line for a week—but still probably undergo some changes. It is also possible that as the sites mature, they will be more stable, but this has not occurred yet.

Finally, we have not dealt with the issue of automatically updating actor dictionaries, depending instead on general international dictionaries that contain country-level information but relatively little information on individual leaders. International news feeds generally include national identification—"United States President Obama," not just "Obama"—so the country-level coding should generally be accurate, but the data probably is less detailed at the sub-state level.

5 Towards a Third-Generation Coder using Open Source NLP[22]

Tabari and VRA-Reader are, in a sense, second-generation automated event coders, as both were based on the *experience* of Keds , the first practical automated coder.[23] As a result of the ICEWS project, several additional proprietary coders have either been developed or are under development: these include LM-ATL's JABARI, a translation (and some debugging) of Tabari into Java, Strategic Analysis Enterprise's XENOPHON, as well as the inclusion of event coders on top of existing software, for example the "Behavior and Events from News" system built into Social Science Automation's *Profiler Plus* and a couple of efforts by BBN and IBM to use their natural language processing software for the task. While one can quibble [endlessly...] on how to mark generations in software, I would argue that these still represent extensions of the second-generation approaches, though in some respects they will overlap with the "third-generation" characteristics I'm arguing for below.

Tabari —and as far as I can tell, the VRA-Reader—are about a decade old, which is a very long time for software.[24] Furthermore, the ICEWS stress-test—while successful in generating data that could be input for models that passed the Phase I accuracy, precision and recall criteria—has revealed a number of issues that have not scaled particularly well to a coding problem of the scale of ICEWS. In the meantime, the overall software environment, particularly with respect to open-source natural language processing, has changed dramatically in the past ten years.

As a consequence, the path towards a "third generation" coder may result in a rather different approach than was used in the second generation. Some of these components are probably already in use, in at least some form, in the new proprietary coders; this section will outline the major steps I see from the perspective of potential open-source coders that could be used in the academic community.

5.1 Use of open-source natural language processing software

Tabari generally operates as a stand-alone system with an internal shallow parser written into the code.[25] Parser code written by a political scientist. It obviously *works*, but the structure of the program is quite complex and pointer-based,[26] the parser cannot be easily

[22]This section in particular has benefitted from a variety of discussions with David Van Brackel, Will Lowe, Steve Purpura and Steve Shellman

[23]There were at least two other efforts to develop second-generation coders, at Maryland and Rice, but neither reached a production stage.

[24]Or not: there are legacy systems that have been going on for fifty years. Like the U.S. air traffic control computers, which date from the mid-1960s despite several unsuccessful attempts at upgrades. A reassuring thought as you read this paper on an airplane, eh?

[25]Tabari actually has the capability of working with input that has been partially marked-up in XML to identify, for example, named entities and pronoun coreferences. This was written for a government contract in the early 2000s but to my knowledge has never been used because the contractor later determined that the mark-up did not significantly improve the coding.

[26]An important speed consideration given the hardware available when the program was written, but not necessarily the best approach now, unless you like code such as `while(*s++ = *pst++);`. Which I do.

updated, it is really hard to find C++ programmers, and it has some well-known quirks, most notoriously matching verb phrase words that are not, in fact, in the verb phrase.

But more to the point, in 2010, unlike the situation in the early 1990s when KEDS was being developed, or even in the early 2000s, the development period of TABARI, when open-source code was still a relative novelty, it makes far more sense to leave the natural language processing (NLP) software development to the computational linguists, and focus only on those remaining tasks that are needed to get convert these structures to events.[27]

Specifically, software for the following tasks can be found at open-source NLP software site such as Open-NLP and various other academic sites:

- Full-parsing. An assortment of full-parsers are available, and the *TreeBank* parse format appears to be a fairly stable and standard output format, so a researcher could use the parser of his or her choice (notably some parser developed in the future) so long as these could produce *TreeBank*-formatted output. At least some of these systems also provide pronoun coreferencing, another feature coded into TABARI. All would provide verb-phrase delineating and delineation of the subject, verb and object phrases of the sentence, the main work required by an event coder.

- Disambiguation by parts-of-speech markup. One of the major tasks of the TABARI dictionaries is noun-verb disambiguation: this issue accounts for much of their size. Parts-of-speech (POS) marking would eliminate this problem.

- Stemming. TABARI has only recently added capabilities of automatically recognizing the regular forms of nouns and verbs. Many NLP systems use stemming—most frequently the Porter stemming algorithm for English. This should both simplify and generalize the dictionaries.

- Named Entity Recognition and Resolution. There is a considerable literature—much of it DARPA-funded—on the recognition of names within text, and also resolving equivalent names, e.g. "President Obama," "President Barak Obama," "United States President Barak Hussein Obama" and so forth. Some of these methods are very sophisticated—for example using conditional random fields and hidden Markov models—and are certainly far more sophisticated than our very simple, if practical when it isn't too slow, *ActorFilter.*

- Synonym and relational dictionaries. The *WordNet* lexical database provides a nearly comprehensive list of synonyms ("synsets"), hyponyms and hypernyms for the English language; this could be used to replace specific instances of nouns and verbs with general classes.

- Sentence detection. As noted above, this is a surprisingly difficult task, and linguists have systems that are more robust than our `perl` filters.

- Regular expressions. Given the ubiquity of regular expressions in the contemporary computing environment—regular expressions have been called "the calculus of

[27]This is presumably the approach that BBN and IBM—which have vast NLP expertise—are doing in developing proprietary coders.

string processing"—it would be very useful to allow these to be used as the pattern-specification language, rather than the ad hoc syntax used in TABARI.

- NGA GEOnet Names Server (GNS). The National Geospatial-Intelligence Agency maintains a continuously-updatyed database of approximately 5.5-million geographical place names (http://earth-info.nga.mil/gns/html/). This could be incorporated into a program for identifying the location of events described in a story which would improve disambiguation of actors and agents. Several commercial systems exist that do this, but we've yet to identify equivalent open-source code, and the application is fairly specialized.

The use of these tools would accomplish at least the following improvements:

- It would align automated event coding—which is fundamentally an NLP problem—with the larger NLP community. As their tools improved, we could incorporate those improvements into our work immediately.

- It would considerably simplify—though not eliminate the need for—the construction and maintenance of a third generation coder, and in particular the tasks that can now be done with open-source ancillary programs would eliminate many of the most brittle parts of the existing TABARI code.

- This introduces a deep—as distinct from a shallow—parser into the system, and the shallow parsing approach has probably reached its limits.

- The use of standardized NLP tools and dictionaries would probably simplify the development of a system for languages other than English, particularly languages such as Chinese and Arabic where considerable NLP work has been invested;

- Many of these features should actually simplify the *.verbs* dictionaries, or at the very least gain more robust performance from dictionaries of the same length;

- While an ambitious prospect, it might be possible to re-define the entire CAMEO coding ontology using the standardized *WordNet* synsets, rather than using the current categories that were developed inductively. This would again help align the event coding with the larger NLP community, and probably simplify its use in languages other than English.

- GNS provides not only names, but coordinates, so this could lead to a very rich set of geo-located data for those events that have an unambiguous location.

Parsing and other pre-processing—in all likelihood a fairly slow process—needs to be done only once for a given sentence, and the marked-up version could be stored, so unlike systems with in-line deep parsers, the resulting coding (which is likely to be re-done many times) should be as fast or faster than the current system. The pre-processing is also trivially divided across multiple processors in a cluster system, so with suitable hardware[28] or using

[28]Of the sort readily available at clod-hopper engineering schools, if not at all defense contractors.

virtual clusters in a cloud computing environment, the processing requirements can be easily adjusted to near-real-time coding environments.

So, what's not to like? Why haven't you already done this?! In a word: dictionaries. The TABARI dictionaries, not the code, actually represent far and away the largest investment of labor in the automated coding, and the existing *.verbs* dictionary has been carefully refined. Adapting these to the new environment is going to involve a lot of effort, though it probably can be done incrementally.

5.2 Hierarchical dictionaries and open code construction

TABARI uses a relatively simple flat-file structure for both the *.actors* and *.verbs* dictionaries, with entries listed in simple alphabetical order. This feature of the system, more than any other, has not scaled well, particularly for the actors. We have consequently used a number of work-arounds—with more radical changes in development at LM-ATL—to get around this, but more systematic approaches are needed.

Changes already mentioned include the automatic code-concatenation with agents and automatic detection of regular noun and verb endings. Other changes involved the ability to keep dictionaries in multiple files, automatic duplicate detection for dictionary entries,[29] the ability of a TABARI *.project* file to call ancillary programs before and after processing, and more flexible formats for specifying actor synonyms and verb forms.

However, we've also moved away from the flat dictionary format, which simply cannot accommodate the 20,000 or so actor phrases we are currently tracking, in contrast to the roughly 1,000 actor phrases in the earlier KEDS work. We began this fairly early—in conjunction with unrelated (except through this file) NSF-funded work on the COW Militarized Interstate Dispute update (Schrodt, Palmer and Hatipoglu 2008) with a general international actors dictionary derived from the *CIA World Factbook* and formatted in an XML-like syntax, e.g.

```
<Country>
<CountryCode>DZA</CountryCode>
<CountryName>ALGERIA</CountryName>
<Nationality>ALGERIAN</Nationality>
<Capital>ALGIERS</CAPITAL>
<MajorCities>
EL_DJAZAIR
WAHRAN
ORAN
</MajorCities>
<Leaders>
<Presidents>
HOUARI_BOUMEDIENNE [19650619 - 19781227]  [B:19320101]  [D:19780101]
RABAH_BITAT [19781227 - 19790209]  [B:19250101]  [D:20000101]
. . .
</Presidents>
<High state committee>
KHALED_NEZZAR [19920114 - 19940131]  [B:19370101]
ALI_HAROUN [19920114 - 19940131]  [B:19270101]
. . .
</High state committee>
<Prime ministers>
AHMED_BEN_BELLA [19620927 - 19630920]
MOHAMED_BEN_AHMED_ABDELGHAN [19790308 - 19840122]  [B:19270101]  [D:19960101]
```

[29] which should have been in earlier versions anyway...

```
ABDELHAMID_BRAHIMI [19840122 - 19881105]  [B:19360101]
KASDI_MERBAH [19881105 - 19890909]  [B:19380101]  [D:19930101]
MOULOUD_HAMROUCHE [19890909 - 19910605]  [B:19430101]
. . .
</Prime ministers>
</Leaders>
</Country>
```

This file is processed by a `perl` program and converted to TABARI *.actors* format, rather than read directly by the program.

This, however, is only a start. In the alphabetical format, different forms a name, e.g.

```
AL-ASSAD [SYRGOV]
ASSAD [SYRGOV]
HAFEZ_AL-ASSAD [SYRGOV]
HAFIZ_AL-ASSAD [SYRGOV]
```

can occur in widely separated places in the dictionary, making maintenance difficult and increasing the possibility that an update will occur on one entry but not all of them.

The solution is to follow basic database principles and move *everything* into a structured file, to that like items such as entity synonyms are all found in a single place. Recent changes in TABARI now allow this at least to the extent of providing synonyms and role/date restrictions in a single block—files with this structure can be read, but not edited, by the program—but most of the dictionaries are still in the older format.

Shellman has recently suggested doing away with the concept of the fixed, free-standing dictionary concept altogether, and simply generate the appropriate dictionary and codes from a general database that contains a wide variety of information on the actor. This would vary depending on the type of actor—for example between individual, state or organization—but might include state and role identities over time, religion, level of militant activity, network connections and the like. Similarly, different coding systems (CAMEO, IDEA, PCS (Shellman 2004), others to be developed) could be incorporated on the same base of vocabulary.

Different elements of this information would be relevant in different applications, and rather than keep all of the information in a single set of codes—our current system, which sometimes results in very long codes—produce customized code containing only the information relevant to a specific application. This is a radical departure from past coding practice—and obviously only possible with fully-automated, high-speed coding—but quite practical in our current environment.

5.3 Duplicate detection and improved content classification

As discussed above, duplicate detection is a major challenge in the current environment. Improved story classification to identify, for example, sports stories, historical chronologies and movie reviews, also would simplify the dictionaries by eliminating the need for a number of discard and null-coded phrases that are present only to avoid coding stories that shouldn't be in the data stream in the first place.

Duplicate detection is a fairly specialized application, and one where we've yet to find much in the way of open source software. However, our sense is that algorithms considerably more sophisticated than those we are using exist in various proprietary aggregation systems,

notably Google News, European Media Monitor, and the academic project NewsBlaster.[30] A more thorough review of the computer science literature might produce some guidance on these issues.

In addition, there is a rich literature with well-documented and robust methods—notably SVM—for document classification, and these may work considerably better than our current keyword-based methods of detecting sports and business stories in particular. There are no technological barriers preventing this, merely the issue of time and money.

5.4 Sophisticated error detection/correction

Thus far, we have been using only limited error detection and correction. Some LM-ATL experiments have shown that even very simple filters can eliminate egregious errors such coding USA/Japanese conflict events based on Pearl Harbor travel and movie reviews or anniversaries of the bombings of Hiroshima and Nagasaki. However, far more sophisticated filtering methods are available, and many of these are of relatively recent vintage due to the computing power required. A multi-category SVM, for example, could be applied to the full text of a story[31] to determine whether the story is likely to have produced events of the type coded, based on previously verified correct codings. The recently-developed "compressive sensing"[32] method for noise-reduction is another method that might be adapted to reduce the presence of incorrectly coded events by determining whether an event is appropriate given the algorithm's assessment of how well it fits with other events in immediate neighborhood.

5.5 Server-based coding

Tabari is designed to run on a single computer. This is still useful, I suppose, if one is working on dictionaries on an airplane, at a typical APSA conference hotel without free internet access, or on a desert island. The first two at least have been know to occur, but in most situations, individuals working on dictionaries have web access, first and foremost to Wikipedia.

The developments outlined above argue for a far more decentralized set of resources—programs and databases—than were found in the integrated environments of the second-generation systems, and suggests—by necessity in some of the potential changes such as automatic dictionary development—moving to a server environment and a web-based interface, probably in Python or php. This solves a variety of problems, including version control when multiple groups are working on dictionaries, operating system incompatibilities, the divergence between the need for a user-friendly interface for some parts of the system (e.g. dictionary development and display) versus high-speed, minimal interfaces for parsing and coding, and multiple language requirements (Java, C++, perl and Python). Servers also simplify the problem of scaling, and allow the possibility of using cloud computing environments, and well as generally moving the system into a 21st-century decentralized computing environment.

[30] Academic but not, alas, open source.

[31] Or possibly a single sentence, but SVMs tend to work better at the document level than the sentence level.

[32] See http://dsp.rice.edu/cs

6 Conclusion

In a history of the first fifteen years of the KEDS/TABARI project (Schrodt 2006), the final section—titled "Mama don't let your babies grow up to be event data analysts" lamented the low visibility of event data analysis in the political science literature despite major advances in automated coding and the acceptance of analyses resulting from that data in all of the major refereed political science journals.

The situation at the present is very different, largely due to ICEWS, which emerged about six months after I wrote that history. As far as I know, all three of the teams involved in the first phase of ICEWS used some form of event data in their models, and LM-ATL, the prime contractor for the only team whose models cleared the out-of-sample benchmarks set by ICEWS, invested substantial efforts in TABARI. Lockheed and various subcontractors have continued to invest in additional developments, both for ICEWS and potentially for other projects, and as noted in the previous section, there are now a number of proprietary systems in active development, in contrast to the previous fifteen years which saw only KEDS/TABARI and VRA-Reader.

Nonetheless, despite the out-of-sample success of the ICEWS models, there still appear to be some barriers in the wider acceptance of the method, largely due to unrealistic expectations about the accuracy of coding—human or machine—and an insufficient appreciation of the ability of forecasting models to deal with noise. In this final section, I will address some of those concerns.

Which, in fact, I also addressed in Schrodt (1994), so little of this is new, but apparently could still use reiterating.

The situation we have with event data is, I suspect, somewhat similar to that faced by George Gallup and other early scientific pollsters in the development of statistical public opinion assessment prior during the period 1930-1970. Three features are of particular note.

First, there were a large number of people who were convinced that they could more accurately assess opinion using unsystematic methods. Notably the infamous political hacks, the pols hanging around the bars near the statehouse chewing tobacco and drinking whiskey and happy to tell any reporter what "their people" really thought.[33] In the not-infrequent instances when these politicians were in fact rigging the election outcomes, they probably were worth listening to, but not as an indicator of true public opinion. But could never be persuaded otherwise—"the pollsters don't know what they are doing" continues to be the refrain of every candidate, anywhere in the world, who is behind in the polls. Likewise, systematic evidence that humans vastly over-estimate their predictive accuracy (Tetlock 2005, of course, but also see http://www.nytimes.com/2010/08/22/business/economy/22view.html?_r=1&scp=2&sq=thaler&st=cse) has been of little avail.

Second, people simply could not grasp the implications of random sampling. The effectiveness of random sampling is, in fact, quite amazing: a properly done sample of about 2,000 people will usually predict the outcome of a national election of 1.3-million voters within a percentage point or so. Extraordinary, really. But then so are iPhones. Gallup had a quick come-back when confronted with the skeptical "How can your poll possibly be accurate: no pollster has ever contacted me!" Gallup's retort: "The next time you go to the doctor for a

[33] For example, that "their Negroes" didn't want to vote at all, but also more broadly.

blood test, don't settle on a little sample, ask him to test all 5 quarts."

We're in a similar situation—but as yet without the clever retort—where despite the demonstrated accuracy of models based on event data, whether in explanatory or predictive settings, people simply cannot believe that a purely statistical model, generated with methods that are 100% transparent, can do better than their anything-but-transparent intuition. And usually demonstrate this by pointing to an incorrectly coded sentence—and any event data system, human or machine, will have plenty of those. But they ignore that fact that the *total* amount of information in the system is vastly greater than that which can be processed by an individual, and while the intuitive analysis may be better in an specific selected case (and certainly for a specific selected news report), the *composite* has better performance. The individual whiskey-swigging pol might do better than a small random sample in their particular ward, or even their city, but what can they say about a city on the other side of the continent? A subject-matter-expert (SME) may perform better on their area of expertise in a particular time frame (though Tetlock's research would suggest not even this is true), but the event-based ICEWS forecasting models predicted five indicators for 29 countries at a monthly granularity for almost fifteen years, and could readily scale this to cover the entire world.

Finally, polls make mistakes, and for example even today it is almost impossible to find a textbook discussion of polling that does not mention the two canonical screw-ups, the 1932 *Literary Digest* "poll" that predicted the re-election of Hubert Hoover, and the "Dewey Beats Truman" forecast of 1948. And there are still forecasting errors, most notably at the moment the failure to anticipate the still-undecided Miller-Murkowski Republican Senate primary race in Alaska.

Yet since the 1970s, most reputable public opinion polls have been correct most of the time—rare is the politician who can boost after the election that "The polls were wrong"—and for that reason a mistake such as Miller-Murkowski is newsworthy in a manner that, say, the errors in the annual political predictions by psychics in the *National Enquirer* are not. Furthermore, because polling methodology—like event data methodology—is generally systematic and transparent, the sources of the errors—typically sampling generally, or incorrect models of voter turnout—can be identified and corrected. In the case of Miller-Murkowski, the problem seems to be a combination of the fact that very little polling is done in Alaska in the first place (it is remote, hard to poll, and doesn't generate a great deal of interest to those who pay for polls compared, say, to California or Texas), and Miller benefitted from the votes of individuals motivated by the "Tea Party" movement who would have been ranked as unlikely to vote in standard models predicting turnout. The latter is a well-understood problem that affects some races, and if someone had wished to expend sufficient funds on Alaska, could have been corrected—or at least reduced—but wasn't. The failure in this one case does not invalidate the method, in fact generally such failures allow for correcting errors and actually make the method more robust in the future.

As I noted in Schrodt (1994), if one is using event data in forecasting models—the objective of ICEWS—coding error is only one potential source of error that lies between "events on the ground" and the predictions of the forecasting model. These include

- News reports are only a tiny, tiny fraction of all of the events that occur daily, and are non-randomly selected by reporters and editors;

- Event ontologies such as WEIS, CAMEO and IDEA are very generic and bin together events that may not always belong together in all contexts;
- Forecasting models always contain specification error and cannot consider everything; for example few if any political forecasting models contain a full economic forecasting component;
- Political systems have a degree of intrinsic randomness due to their inherent complexity, chaotic factors even in the deterministic components of those systems, the impact of effectively random natural phenomena such as earthquakes and weather, and finally the effects of free will, so the error intrinsic to a forecasting model will never reduce to zero.

In this chain of events, the impact of coding error in automated systems, while still relevant, is not necessarily dominant. The first and fourth factors also affect SME evaluations; the second and third affect models based on human coding. And the bottom line is that in gold-standard, out-of-sample predictive tests, models using event data work.

When assessing the alternative of human coding, there are two additional problems. The first is the simple impossibility of human coding. TABARI can code 26-million records in 6 minutes, resulting in about 3-million events. Sustained human coding projects, once one takes in the issues of training, retraining, replacement, cross-coding, re-coding due to effects of coding drift and/or slacker-coders and so forth, usually ends up coding about six events per hour.[34] The arithmetic is obvious: 6 minutes of automated coding, or 500,000 labor-hours of manual coding, probably costing on the order of $10-million when labor and administrative costs are taken into effect. And that's to code the texts once.

For this reason, human-machine comparisons are of little practical consequence, since human coding is not an option. Multiple independent tests, most recently King and Lowe (2004) have shown that machine coding is comparable in accuracy to human coding.[35] But the human coding accuracy in some of those tests is quite low: King and Lowe use an assortment of measures (and a fairly specific sampling method) but the accuracy on the individual VRA codes alone (Table 2, pg 631)—not the complete record with source and target identification, another major potential source of error—is in the range 25% (!) to 50% for the detailed codes and 55% - 70% for the cue categories.

Unfortunately, we don't have a contemporary large, randomly sampled human coded comparison data set—given the futility of human coding as an alternative to automated coding, no one has invested the very substantial amounts of time and money that would be required to do this,[36] but based on my experience and anecdotal reports from various other event data coding projects (Maryland's GEDS, the CACI project for the NSC 1981-1985, Third Point Systems for the Saudis in the 1980s, Russ Leng's BCOW at Middlebury)

[34]Individual coders, particularly working for short periods of time, can reliably code much faster than this. But for the *overall* labor requirements—that is, the total time invested in the enterprise divided by the resulting useable events—the 6 events per hour is a pretty good rule of thumb and—like the labor requirements of a string quartet—has changed little over time.

[35]See also Schrodt and Gerner 1994.

[36]The *major* problem with such an exercise is reaching convergence among the human coders: about ten years ago VRA undertook a substantial, well-designed exercise to do this but no results ever came of it, apparently because the coding never came close to a consensus.

over the years, that sustained accuracy is will be in the range of 70% at best. The human-coded COPDAB data set somehow manages to miss the Korean War (Howell 1983), the human-coded GEDS project, which consumed to bulk of the event data expenditures of the NSF-funded "Data Development in International Relations" project has not been used in a single refereed article.

This is not to say that continued efforts should not be made to improve the quality of event coding, and I outlined in the previous section a variety of relatively low-cost (and not so low-cost) ways that this could be done. *Kaizen* is a really good idea, particularly when obvious advances are clearly available. Returning to the survey research analogy, we see this as well: voter-turnout models are continually refined, and at times major adjustments need to be made, as in the trend to replace easily sampled land-lines with cell phones. Not all of the methods that worked for the focused event data sets of the KEDS project will work for the much more comprehensive data required of ICEWS and potential successors to ICEWS.

But the utility of event data should be assessed on results. In 1962, Deng Xiaoping famously quoted the Sichuan proverb, "No matter if it is a white cat or a black cat; as long as it can catch mice, it is a good cat." In the short term—the Cultural Revolution being just over the horizon—probably not the brightest thing to say; in the long term, it became symbolic of the road to China's transformation into a major world economic power.

Statistical forecasting models utilizing event data coded with automated techniques are good cats. Some are white, some are black, but they catch mice. Furthermore, the fact that such models exist is now known—O'Brien (2010)—and from a policy perspective it is likely that they will be continued to be developed for policy applications seems rather high: the open-access textbook on the results of the KEDS project circa 2000, *Analyzing International Event Data,* has been translated into Chinese.[37] The cat, so to speak, is out of the bag.

Furthermore, while there is certainly a place for proprietary code, the existing work has advanced rapidly due to very significant synergies with open-source software. This has occurred both between NSF-funded research and contractual DARPA-funded research, as well as the use of open-source resources in computational linguistics (much of this also DARPA funded). This accelerated development has not begun to slow. ICEWS has presented a positive case-study in the use of integrating open-source software into a major policy-oriented project—apparently a general trend in the field[38]—and with additional investments still forthcoming in that project, the immediate future of event data looks very positive.

[37]Or so I've been told, though I've not been able to locate this on the web. Itself interesting.

[38]See http://www.wired.com/dangerroom/2010/08/cia-software-developer-goes-open-source-instead/. Note that report specifically identifies Lockheed-Martin as one of the good guys in this regard.

References

[1] Azar, Edward E. 1980. "The Conflict and Peace Data Bank (COPDAB) Project." *Journal of Conflict Resolution* 24:143-152.

[2] Bond, Doug, Joe Bond, Churl Oh, J. Craig Jenkins and Charles Lewis Taylor. 2003. "Integrated Data for Events Analysis (IDEA): An Event Typology for Automated Events Data Development" *Journal of Peace Research* 40,6: 733745

[3] Gerner, Deborah J., Philip A. Schrodt, Ronald A. Francisco, and Judith L. Weddle. 1994. "The Machine Coding of Events from Regional and International Sources." *International Studies Quarterly* 38: 91-119.

[4] Deborah J. Gerner, Philip A. Schrodt and Ömür Yilmaz. 2009. *Conflict and Mediation Event Observations (CAMEO) Codebook*. Manuscript, `http://web.ku.edu/keds/data.dir/cameo.html`

[5] Goldstein, Joshua S. 1992. "A Conflict-Cooperation Scale for WEIS Events Data." *Journal of Conflict Resolution* 36: 369-385.

[6] Howell, Llewellyn D. 1983. A Comparative Study of the WEIS and COPDAB Data Sets. *International Studies Quarterly* 27: 149-159

[7] Leng, Russell J. 1987. *Behavioral Correlates of War, 1816-1975.* (ICPSR 8606). Ann Arbor: Inter-University Consortium for Political and Social Research.

[8] King, Gary and Will Lowe. 2004. "An Automated Information Extraction Tool for International Conflict Data with Performance as Good as Human Coders: A Rare Events Evaluation Design." *International Organization* 57,3: 617-642.

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

[10] McGowan, Patrick, Harvey Starr, Gretchen Hower, Richard L. Merritt and Dina A. Zinnes. 1988. International Data as a National Resource. *International Interactions* 14:101-113

[11] Merritt, Richard L., Robert G. Muncaster and Dina A. Zinnes, eds. 1993. *International Event Data Developments: DDIR Phase II*. Ann Arbor: University of Michigan Press.

[12] O'Brien, Sean. 2010. Crisis Early Warning and Decision Support: Contemporary Approaches and Thoughts on Future Research. *International Studies Review* 12,1:.87-104

[13] Schrodt, Philip A. and Deborah 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.

[14] Schrodt, Philip A. 1994. Statistical Characteristics of Events Data. *International Interactions* 20,1-2: 35-53.

[15] Schrodt, Philip A. 2006. Twenty Years of the Kansas Event Data System Project. *The Political Methodologist* 14,1: 2-8.

[16] Schrodt, Philip A. 2009. TABARI *: Textual Analysis By Augmented Replacement Instructions*. PDF file, `http://web.ku.edu/keds/tabari.html`. (accessed 10 August 2009)

[17] Schrodt, Philip A., Glenn Palmer and Mehmet Emre Hatipoglu. 2008. Automated Detection of Reports of Militarized Interstate Disputes Using the SVM Document Classification Algorithm. American Political Science Association, Boston.

[18] Philip A. Schrodt, Deborah J. Gerner, and Ömür Yilmaz. 2009. Conflict and Mediation Event Observations (CAMEO): An Event Data Framework for a Post Cold War World. Jacob Bercovitch and Scott Gartner. *International Conflict Mediation: New Approaches and Findings*. New York: Routledge.

[19] Schrodt, Philip A. and Deborah J. Gerner. 2004. An Event Data Analysis of Third-Party Mediation. *Journal of Conflict Resolution* 48,3: 310-330.

[20] Schrodt, Philip A. 2009. Reflections on the State of Political Methodology. *The Political Methodologist* 17,1:2-4.

[21] Shellman, Stephen M. 2004. Measuring the Intensity of Intranational Political Interactions Event Data: Two Interval-Like Scales. *International Interactions* 30,2: 109-141.