Coding with TABARI is a balancing process. Almost every aspect of good coding walks a line between opposite pitfalls: phrases that are too specific and phrases that are too general; spending too little time on a lead and giving up on it after ten seconds; trying to code for five hours straight and coding for two hours per week. The trick is to stay between the extremes.

It is helpful when coding to keep the long-range goal in mind. With TABARI, coding by hand is a means, not an end. Hand coding[1] is done only to develop the dictionaries that the program runs on; no useful data is generated until the dictionaries are complete and the program autocodes all of the leads over again. Therefore, the point of dictionary development is development, not coding individual leads correctly. Extracting a useful actor or verb dictionary entry from an uncodable or incorrectly coded lead is just as useful as getting an individual lead to code correctly. Additionally, even leads that are coded correctly can be used to extract useful actors and phrases to add to the dictionaries.

This fact is both very freeing and very sobering. It means that any single lead does not so much matter whether it codes correctly or not; the coder need not worry over the inevitable "bad leads" he or she runs into. On the other hand, this also means that correctly coding a single lead accomplishes nothing useful if it requires changing the dictionaries in ways that will cause future leads to miscode. Every potential dictionary alteration must be evaluated as to whether it will cause more harm or more good when it is applied across thousands of leads.

Before diving into the assorted problems that coders face, it should be pointed out that TABARI is a very good program. Simple leads and straightforward events get coded correctly the vast majority of the time.[2] For this reason, our primary attention is focused on those cases where regularly occurring structures in leads cause TABARI to systematically miscode. A focus on the inevitable occasional problem should not cause despair; when confronting a problem lead it is easy to forget that one just finished breezing through ten correct leads in a row.

Meditations on the Dictionaries

The coder's task is to add entries to four lists of words and phrases, called dictionaries. The two primary ones are the ACTORS and VERBS dictionaries. The NOUNS and ADJECTIVES dictionaries are supplementary lists used to weed out common problem words and phrases.

The Good The simple structure of the actors dictionary—an alphabetized list of names—is reflected in the simplicity of using it. The coder, so long as he or she is sober, need not worry much about causing problems by adding to the actor dictionary. The most likely trap the coder is likely to encounter with the actor dictionary is not adding to it

[1] By "hand coding" is meant the process of dictionary development by human coders to prepare TABARI for autocoding the entire collection of leads. In the context of this document, "Hand coding" does not refer to the use of human coders to manually develop event data without the use of a machine coding program. That would be nasty, brutish, and painfully long.

[2] That is, after the dictionaries have reached a reasonably advanced stage

enough. TABARI's conjunction-recognition feature and the way it searches for actors make it desirable that all permutations of a leader's country, title, and name be included as separate entries. For example, KING_ABDULLAH, KING_ABDULLAH_II, and JORDAN'S_KING_ ABDULLAH, should all be separate listings in the actor dictionary with the code [JORGOV]. It is tempting when coding to see that the phrase "Iraqi Prime Minister Tariq Aziz met with Yasser Arafat" correctly generates the event IRQ 022 PALPLO (from IRAQI MET WITH YASSER_ARAFAT) and move on to the next lead, but the coder needs to recognize that it ought to generate IRQGOV 022 PALPLO and add IRAQI_PRIME_ MINISTER_TARIQ_AZIZ [IRQGOV] to the actors dictionary. Adding multiple spellings and arrangements of a name risks no miscodes and offers the potential for better interpretation of leads.

The Bad There are no bad TABARI dictionaries, only bad TABARI coding schemes…[3]

The Ugly Verb phrases are more complicated. There are only a limited number of ways a person's name can be written (though the number of different ways journalists refer to "Lebanon's Shiite Muslim fundamentalist militia Hezbollah" is astounding), but the English language's flexibility allows for much more variation in wording to describe the same event. For example, "met," "consulted," "lunched with," "arrived for talks with," "visited," "headed to," and "hosted" are just a few of the ways that meetings between heads of state are indicated. Additionally, the same word can mean very different things; "fired" can mean a termination of employment or a military engagement.

This creates the quandary that the verbs dictionary must be large and contain many patterns, while at the same time the coder must be more cautious about changes made to the verbs dictionary than to the actors dictionary. When creating a verb entry, two questions must be asked. First, will the entry result in a correct coding of the lead in question (making the assumption that the current lead is fairly representative of future leads)? Second, could the entry create miscodes if a word in it is used in a different way in the future? The verb "attacked" by itself cannot be coded as [190] "use conventional military force" because it is often used to describe verbal criticism, as in "attacked the Russian policy." Instead, longer phrases including the verb must be separately entered, such as TANK ATTACK [194] and ATTACK IN_ SPEECH.

The problem of multiple interpretations tends to decrease as the length of a verb phrase entry increases, but the danger still needs to be considered. Unfortunately, using only long verb phrases introduces a different problem: applicability and recurrence. The point of machine coding is to work from a list of common words and phrases. Using only long verb phrases will allow a coder to correctly treat individual leads, but long phrases are less to appear in future leads, and thus the effort in creating them is wasted. A short, common phrase but one that excludes incorrect interpretations is the goal of a verb dictionary entry.

[3] See "SFP"

Verb phrases, then, are another example of the balancing act in coding. Walking the line between phrases too short and too long, too general and too specific, is the task at hand. A good way to approach the creation of a verb phrase is to take the lead and "boil it down," letting extraneous words and phrases evaporate until only a core of the lead that corresponds to one and only one CAMEO category remains.

For example, take the lead "Israeli Prime Minister Ehud Barak headed here Sunday to discuss the Middle East peace process with Egyptian President Hosni Mubarak, whose top advisor said a Syrian-Israeli settlement could take a year." ISRAELI_PRIME_ MINISTER_EHUD_BARAK is clearly the source; this makes these words unlikely to be useful for inclusion in the verb phrase. The words "here" and "Sunday" are irrelevant to the CAMEO coding scheme, so they can be dropped. The opinion of Mubarak's "top advisor" is extraneous to the event of the meeting and also not of much significance to the CAMEO scheme, so it can be dropped as well. This leaves us with "ISRAELI_PRIME_ MINISTER_EHUD_BARAK headed ~~here Sunday~~ to discuss the Middle East peace process with Egyptian President Hosni Mubarak, ~~whose top advisor said a Syrian-Israeli settlement could take a year~~." The target is now clear as EGYPTIAN_PRESIDENT_ HOSNI_MUBARAK. Closer consideration of the CAMEO scheme reveals that the topic of discussions is irrelevant; what we are interested in is the fact of the meeting. "The Middle East peace process" can thus be discarded.

Now we have "ISRAELI_PRIME_ MINISTER_EHUD_BARAK headed ~~here Sunday~~ to discuss ~~the Middle East peace process~~ with EGYPTIAN_PRESIDENT_HOSNI_ MUBARAK, ~~whose top advisor said a Syrian-Israeli settlement could take a year~~." There isn't any extraneous information left; the lead has been boiled down to its essence, which we see as "ISRAELI_PRIME_ MINISTER_EHUD_BARAK headed to discuss with EGYPTIAN_PRESIDENT_HOSNI_MUBARAK." Substituting our TABARI symbols, we have "$ headed to discuss with +". This is our verb dictionary entry. We use the first verb in the phrase as the anchor,[4] since TABARI uses the first verb it encounters in the lead as the preferred one. We enter "* TO_ DISCUSS WITH_ +" in the verb dictionary under the verb HEAD.[5]

Now all that remains is to assign the phrase a CAMEO code. In assigning the code, our primary reference is the verb dictionary entry itself, *not* the lead in its entirety. We are developing the dictionary, not coding the lead, and the dictionary entry will remain long after this particular lead is forgotten. This means the CAMEO code must be derived from

---

[4] The verb heading in the dictionary under which the verb phrase in entered

[5] Notice that the $ operator has been left off. This is because TABARI assumes without being told that the source comes before the verb. TABARI searches for "$ * +" as the default pattern with respect to every verb, and when other words are added to a verb phrase it continues to assume that the source comes first unless it is ordered otherwise in the phrase. It normally accomplishes nothing to include the $ as the first character in a verb phrase. $ is primarily intended for phrases where the source does not come first, as in "+ was killed by $." One exception to this rule is when an entry is intended to force the source to be in a specific location with respect to another word, as in "$_WARPLANES *" under the verb RAID. This would prevent a lead like "In the Gaza Strip Israeli warplanes raided office buildings that the Israeli military said housed Hamas forces" from being coded as PALGZ RAID ISRMIL. Instead, it would be coded ISRMIL RAID PALHM.

the text of the verb phrase, not from the lead itself.[6] Our lead above refers to discussions, so it will fall under the CAMEO category for "Consult," the 020 category. More specifically, our phrase uses the verb HEAD, indicating travel, so the phrase receives the code 022, "Make a visit." Since visits are always hosted by someone, the 022 code is paired with the 023 code, "Host a visit." Our complete verb dictionary entry thus looks like this:[7]

HEAD
- * TO_ DISCUSS WITH_ + [022:023]

Clearly, adding verb dictionary phrases is much more involved than simply typing in the name of a new actor, or a new combination of title and name. With practice, though, the process becomes intuitive, and the coder soon loses consciousness of going through each individual step.

The Others The actor and verb dictionaries are used to generate events. In contrast, the noun and adjective dictionaries are used to prevent the generation of incorrect events, and to prevent useful leads from being thrown out. There are two ways in which they do this. The simpler way is by providing a place to file words that look like verbs or actors but really aren't. For example, "Middle East peace process" is a common phrase that would trigger the actor MIDDLE_EAST, as in "Anthony Zinni traveled to the Middle East today." The "Middle East peace process," though, is neither a destination nor an actor—just a theoretical concept. By adding MIDDLE_EAST_PEACE_PROCESS to the noun dictionary the phrase is ignored as a noun—a noun cannot simultaneously be an actor—and the problem is avoided. (This process is even more important regarding false verbs, but that will be discussed later.)

The second function of the noun and actor dictionaries (and the reason that they are separate lists) is the weeding out of false clause conjunctions. TABARI is set up to consider conjunctions like AND and OR as the boundaries between separate parts of a compound sentence. "President Clinton met with Binyamin Netanyahu yesterday AND the Israeli Prime Minister agreed to open negotiations with the PLO" is an example of such a sentence containing two events, USAGOV MET WITH ISRGOV and ISRGOV AGREED TO NEGOTIAT WITH PALPLO. However, AND is also often used to join two things together that are not separate clauses, as in "President Clinton met on this bright and sunny day with Binyamin Netanyahu." The danger here is that TABARI will interpret the AND to mean there are two separate sentences. Since neither side of the lead

[6] Of course, a well-crafted verb phrase will capture the essence of the lead in such a way that the code for the two will be identical. This is not true absolutely because sometimes the idiosyncrasies of wording in a lead will suggest something that a verb phrase cannot capture. In these cases, preference should be given to the text of the new verb phrase over the text of the lead in assigning a code, again because the verb phrase is what will endure, not the lead.

[7] Those familiar with the CAMEO scheme will note that the phrase might suggest the code 024:024, "Meet in third location." In this case 022:023 is preferable because there was no location given to suggest a site outside of Egypt. The phrase "* TO_ ^ TO_ DISCUSS WITH_ +" would be a better candidate for 024:024, although even this is not certain because a lead could say "Israeli Prime Minister Ehud Barak headed to Cairo to discuss the Middle East peace process with Egyptian President Hosni Mubarak."

has two actors, it will not be able to generate a $ VERB + structure, and it will conclude, "No events found." The event data from the lead is thus lost.

With the adjective dictionary, this problem is solved. TABARI has a feature that recognizes structures that consist of the sequence ADJECTIVE_AND_ADJECTIVE or NOUN_AND_NOUN.[8] These structures are then treated as parts of a single clause rather than the border between two separate ones. If the coder adds "bright" and "sunny" to the adjective dictionary, TABARI will read "CLINTON MET on this adjective_and_adjective day with NETANYAHU" rather than "CLINTON MET on this bright AND sunny day with NETANYAHU," which is useless. The noun and adjective dictionaries, then, offer a "bridge" over troubled conjunctions. As it turns out, this bridge is useful in a remarkable number of leads.

The noun and adjective dictionary features are relatively recent additions to TABARI, and so right now they remain rather small. However, they are powerful tools. The conjunction problem was one of the biggest sources of systematic errors in TABARI coding of newswire leads, and these two dictionaries go far toward minimizing it. The coder simply needs to remember to use the function, and to remember that TABARI only recognizes the strict format NOUN_AND_NOUN or ADJECTIVE_AND_ADJECTIVE. This means that occasionally multiple-word phrases must be used to trigger recognition, such as APPEALS_COURT. In general, creative use of nounizing and adjectivizing is a good thing, so long as consideration of what potential problems might be created by doing it is retained.

A Painfully Detailed Look at the Coding Process

Having plumbed the depths of the dictionaries and delved deeply their mysteries, it is time to consider the process that creates them, bit by bit extracting useful names and phrases from the raw material of daily newswire reports. "Coding," it is called, that curious activity that exposes one to the mundane details of what happened each day during a period several years ago. At times quite enjoyable, at times frustrating, at times simply slow, human hand coding is fundamental to the machine coding project. Ironically, it turns out that human coders and the TABARI program have something in common: neither of them can continue coding for too long a sustained period.[9]

---

[8] There are separate dictionaries for nouns and adjectives rather than just throwing all the words together into one big list because it is not useful to search for NOUN_AND_ADJECTIVE or ADJECTIVE_AND_NOUN; these structures, if they occur, do not usually need to be bridged. The exception to this—and a point that may allow future addition to the program—is the structure NOUN_AND_ADJECTIVE_NOUN, as in "the president and two advisors."

A variation of this structure is the sequence ACTOR_AND_TITLE_ACTOR, as in, "French President Jacques Chirac and Prime Minister Tony Blair." Here the intervening "Prime Minister" will interrupt recognition of the compound actor JACQUES_CHIRAC_AND TONY_BLAIR unless PRIME_MINISTER_TONY_BLAIR is a separate actor dictionary entry from TONY_BLAIR—once again highlighting the need for extensive inclusion of name and title variations in the actors dictionary.

[9] Of course, this is for two very different reasons. Human coders cannot code for too long at a stretch because their brains tire and attention spans run out, and they begin to fall asleep, watch the clock, compulsively check their email, and daydream. TABARI cannot code for too long at a stretch because it rapidly completes all the leads it has been supplied with and then has no material on which to continue coding. Still, the statement that the two are similar stands because in the literal sense it is true, and in the

For the human coder, the basic approach to attacking a lead has four steps:

1. Read the sentence from beginning to end.
2. Mentally pluck from among the extraneous words in the sentence the essence of what is being reported. What actually happened that can be translated into event data, as opposed to editorial commentary, prognostication, or useless human-interest detail? Some leads will contain nothing useful; others, multiple pieces of information that can be turned into event data. When something useful is found it will take the form ACTOR ACTION ACTOR (at least within the WEIS or CAMEO framework), where one entity is doing something to another entity.[10]
3. Having extracted the essence of the lead, turn to the TABARI output line, which
   will show how the program has interpreted the text. Conveniently enough, the TABARI output will also exist in the format ACTOR ACTION ACTOR, where each actor is a represented by a three- or six- letter sequence and the action is represented by a three or four digit number, followed by text to describe what that number represents. Do the essence of the lead and the TABARI interpretation match? If they do, pat the monitor on the top and move on to the next lead.
4. However, often they will not match, or an error code will be generated. This prompts a quick diagnosis as to whether the lead is (a) worth correcting and (b) correctable. If both of these are the case, a trial-and-error session is initiated, where entering new words and phrases into the actors and verbs dictionaries hopefully results in the correct coding of the lead. As has already been seen, actors can be entered without a second thought; they are straightforward and rarely cause recognition or stemming problems. Verb phrases are more difficult; it may take several tries to find an accurate verb pattern that the program will match to the lead. Don't hesitate to try several different dictionary changes while searching for one that will work. Failed changes can always be deleted without causing any harm, and sometimes a new entry that does not fix the current lead might be left in the dictionary because it seems likely to be applicable to future entries.

With the general picture of coding established, a more detailed look at typical coding scenarios is next to consider. The step-three comparison of a lead with TABARI's interpretation of it usually yields one of four situations: a perfect match, an almost match, a waytysm match, or an error code.

---

literary sense drawing such comparisons is all about justifying the most unlikely statements on whatever grounds possible, level of flimsiness notwithstanding.

[10] The KEDS project uses the word "actor" to refer to a person, country, or organization that can initiate or receive an action: visits, criticisms, economic aid, etc. In a particular lead, an actor will be either the SOURCE ($) or the TARGET (+), depending on whether it is doing the action or receiving it. An action is an incident between two actors that is defined by the CAMEO codebook and referred to by a number, called a "code."

Perfection The most desirable of these, obviously, is a perfect match. If TABARI nailed the lead—not only correctly extracting the event described, the source, and the target but also extracting *all* events in the text and not adding in any extra ones—then the lead can be passed up for the next one. In advanced stages of a coding project, after the actor, verb, noun, and adjective dictionaries have been well developed, perfect matches are quite common, though certainly not universal.

Nobody's perfect The second possibility is an almost match. This is a situation where the program has done well, but conceivably could have done better. Almost matches are sometimes close enough and sometimes not, but regardless of how "almost" the result is, the coder should at least briefly try to improve it. Examples of almost matches are instances where the actors are both correct but the event code is not—either it is completely wrong or it is semi-right but there is a more appropriate code available.[11] Alternatively, the event code may be correct but one of the actors is incorrect or missing.

Almost matches can be relatively simple to fix. If the actors are both correct, simply adding a new verb pattern will almost always fix the problem. It may be that a short verb pattern is being picked up inside a longer phrase that has a much different meaning: "said will comply" versus "said will not comply." Simply adding the longer phrase will solve the difficulty. If the actors are not correct, one of them may be missing from the dictionary altogether; adding it will fix the problem. Sometimes an event will be generated between a state and that state's government—ISR and ISRGOV, for example. Often, these will be cases where a politician and his or her title are being picked up separately. "Israeli prime minister Ariel Sharon…" may be coded as an event between ISRAELI and ARIEL_SHARON. Here, adding the new actor phrase ISRAELI_PRIME_MINISTER_ ARIEL_SHARON will fix the mistake.

One last type of almost match is the classic reversal of source and target: Palestinian protestors arresting Israeli police. The latest versions of TABARI greatly reduce the incidence of actor reversal, but it still occurs with some frequency. Simply adding a "+ * BY_ $" (or similar) phrase in the verb dictionary will usually solve the problem.

When the world falls down around you The third type of match is the "what are you thinking, you stupid machine" match: waytysm. These matches are usually generated by leads with unusual or complicated grammatical structures or by idioms or direct quotes. Additionally, there are systematic newswire lead devices that generate waytysm matches, the most common being the "So-and-so did thus and such, said $" structure. The COMMA SOURCE SAID sequence at the end of the lead gets dropped by TABARI's feature that eliminates subordinate clauses and then the source's opinion, command, or interpretation gets coded as an actual event.

---

[11] A semi-right code, in the light of the statistical application of event data, is probably good enough most of the time, especially if it is in the same category (02, 05, etc.) as the desired code. Still, in the pursuit of the best data possible, in view of future leads that may require a new phrase to be even remotely correct, and in view of the negligible effort required to do so, it is a good idea to try to get the code perfectly correct by adding a new verb pattern.

Waytysm leads quite often are simply beyond repair. If the problem is a structural one, as in the example above, there is little a coder can do. Sometimes a useful verb dictionary entry can be extracted from the lead so that this particular lead may be miscoded but future similar leads might be correctly interpreted. Other times, the word sequence that creates the problem can be entered and null-coded, or given a useless code like "make neutral comment." In this way, the lead is not coded correctly, but it is not coded incorrectly either. Removing a source of problems is useful, though not nearly as satisfying as adding a phrase that leads to correct coding.

Sometimes a waytysm lead happens simply because a new type of event is reported. Having never been seen before, the event is incompatible with the information the dictionary has been built upon. These types of problems are easy to solve; simply creating the necessary language will do the trick. Sometimes this may be as simple as adding a new verb to the dictionary.

One possible outcome of a confrontation with a waytysm lead is an epic struggle between TABARI and the coder with a happy ending. Occasionally, wrestling with a lead and making five trial changes to the dictionary that are undone followed by eight lasting changes results in a difficult lead being correctly (and oh-so-elegantly) coded. There is a drawback to this scenario. One lead, even changed from grossly miscoded to nailed-tight correctly coded, is still only one lead. In the statistical analysis of the data, it will not show up, and there is a risk that such radical dictionary surgery will result in ten leads that were previously coded correctly now being butchered. Also, the investment of fifteen minutes on a single lead is simply not economical, and if the resulting verb patterns added to the dictionary are long and complex, they will never help in the coding of another lead. This defeats the purpose of dictionary development and machine coding.

On the other hand, the epic struggle does have some benefits. For the novice coder, the epic struggle may actually involve a straightforward lead, and the process of working through it yields insight into how TABARI works, how leads get coded, and what works and what does not in the dictionaries. This is learning by doing, and after a few repetitions the epic struggle may shrink to a ten-second surgical strike that epitomizes the purpose of hand coding. For the advanced coder, other benefits accrue. Someone who knows what they are doing will create dictionary entries that, though they may be fairly long, contain common words and that are applicable to future leads. There are also significant morale advantages to an epic struggle,[12] both from personally conquering a twisted-grammar Matterhorn and from the future sense of awe that bringing up a new, complex lead that TABARI nails will bring to some fortunate coder. Further, the experienced coder will conduct his or her war against the dictionary in such a manner as to not initiate the miscoding of hundreds of other leads in order to address the eccentricity of one. Considering the benefits and risks, then, the key word regarding the epic struggle is (surprise, surprise) "balance."

[12] Though probably not of the sort that could inspire a blockbuster movie and make one wealthy and famous for having done very little.

In short, the waytysm lead can be frustrating, but it offers an opportunity to learn and to glean useful phrases for the verb dictionary. A rational approach where the coder gives the lead a good shot while refraining from obsession with "solving" it is the goal. No matter what the outcome of that particular lead is, there are hundreds more following it. This depressing thought serves to highlight again the purpose behind the whole exercise: dictionary development. If useful dictionary entries are extracted, all is well. It's like working out; how far a basketball player runs or how much weight he or she lifts in any particular practice session is irrelevant as long as the outcome is better skill during the game (though there is usually a correlation between them…).

Two last thoughts about waytysm miscodes (and miscodes in general). First, a certain number of problems are the direct result of features of TABARI—the way it interprets a type of word (like verb, conjunction, actor, etc.), the way it addresses punctuation, the order in which it searches for events. In diagnosing a problem like this, it is crucial that the coder understand how the program works: what it looks for, what it ignores, and the order in which it searches. This allows the quick recognition of which problems are fixable and which are beyond hope. Just as how people can often predict how their friends will react to a certain situation, the coder should be able to make a good guess at how TABARI will interpret a lead even before seeing it done. Essentially, the coder must think like the computer. Developing the ability to turn off one's human flexibility—the inherent knowledge, contextual understanding of sentences, and intuition about how things ought to work—and instead process a sentence in the mechanical, rigidly systematic, senseless way that a computer program does is crucial to figuring out why some leads work out right and why others just don't cut it. This, in turn, allows the coder to turn his or her creativity back on to think of a solution to the problem. Once again balance is crucial, this time between mechanical processing and creative problem solving.

Second, even errors that are the result of TABARI's structure are not necessarily "forever problems." If a regular structure causes the same problem over and over, an alteration of the program itself might be able to fix it. For this reason TABARI includes the "Problem" feature under the "Options" menu. Use it! Selecting this feature allows coders to type a description of whatever difficulty they are encountering. This description is then saved to a separate word-processing file outside of the program, along with the current lead and the parsing of that lead. These problem files can then be forwarded to a project leader. Over time, an accumulation of various problem files can suggest potential, which then can be incorporated into TABARI's source code itself—though this obviously requires a good programmer. Accumulated experience and familiarity with common problems was what allowed Dr. Schrodt to incorporate many of the most recent features of the program, including the nouns and adjectives dictionaries. Outside the level of programming changes, problem files can also be used to highlight difficulties with the coding scheme or simply as a convenient way of recording questions about how to code certain words or types of events.

<u>When the computer throws up its hands in despair</u> The three types of matches are common, and the coder will do much work with them. However, sometimes TABARI does not even reach the point of trying to code the sentence—it just gives some lame

excuse about why the lead is "just too hard" and "it makes my chip hurt" and begs to be allowed to skip on to the next one. In these cases where no match is generated, the coder has to do triage:

1. Assess the type of problem from the error code generated
2. Determine whether this type of problem is fixable
3. Fix it or move on

There are several different reasons TABARI might refuse to code a lead. Some are valuable. Discard codes, for example, are words or phrases that are entered by the coder into the actor dictionary to trigger the immediate, perfunctory dismissal of any lead that contains them. This allows the efficient discarding of leads about sporting events and presidential debates, as well as more idiosyncratic events like the spilling of massive amounts of cyanide into the Tisza River in Romania that killed thousands of fish and contaminated downstream into Hungary in early 2000. Other examples of useful non-codes are leads that do not contain a verb, or leads that do not contain both a source and a target.

Other times, though, a non-code is not the best outcome. Sometimes the coder can fix this, most commonly in the "too many verbs" case. TABARI is set up to ignore any lead that has six or more individual words with the "verb" tag. Presumably, any lead containing six verbs is too complicated for the program to code correctly, so it refrains from even trying. However, TABARI often incorrectly labels nonverbs as verbs. If a coder can "deverb" the false verbs in a lead, he or she can bring the lead below the verb limit[13] and make it codable. This has the additional benefit of making future leads that contain the same false verbs codable as well. The cumulative effect of deverbing one or two words in a lead over hundreds of leads can be enormous, allowing the coding of many leads that would have otherwise been lost to the verb limit.

False verbs often arise from the stemming feature of the program. For example, if the verb "sold" is in the verb dictionary, the words "solder" and "soldiers" will be identified as forms of that verb. Even ignoring the verb limit, this is problematic because these words will trigger the recognition of verb patterns that are not accurate, leading to miscodes. While a little caution in the creation of new verb entries can help prevent such problems (will "sold_" work just as well as "sold"?), some stemming is necessary for the recognition of –ed, -s, and –ing forms, and occasional unforeseeable problems arise as well. Basically, some false verbs cannot be systematically prevented—desperate measures are required. A classic example of problematic stemming is the AFP lead

> Sharon's apparent turnaround came after Israel sent its tanks and helicopters into the Gaza Strip and the West Bank, resulting in heavy clashes with gunmen and scenes of carnage, and ahead of new missions to the region by a US Middle East peace envoy and Vice President Dick Cheney.

[13] Parallelism here suggests that the word "verbing" be used to describe the use of too many verbs. Of course, it would also suggest a fine for such an incident, so maybe that is not tbe best idea…

A lead of this complexity is exactly what the verb limit is designed to throw out. It does not even report a "hard" event that can be coded, just an "apparent turnaround." However, there are not enough verbs to trigger the limit, so the lead is coded

020308	ISR	PALGZ	014	(ACCOMODATE, CEASEFIRE)
020308	ISR	PALWB	014	(ACCOMODATE, CEASEFIRE)

A ceasefire! It's certainly good that we have TABARI to interpret the news for us.

Closer analysis reveals the problem.

> Sharon's apparent **turnaround** came after Israel sent its tanks and helicopters **into** the Gaza Strip and the West Bank, resulting in heavy clashes with **gunmen** and scenes of carnage, and ahead of new missions to the region by a US Middle East peace envoy and Vice President Dick Cheney.

The phrase "* IN GUN" is entered in the verb dictionary under the verb TURN, and turning in weapons is a form of accommodation. Thus "turnaround" is recognized as the verb, "into" as the preposition "in," and "gunmen" as the word "gun."

The error makes perfect sense, and it is easily fixable through the desperate measures of the noun and adjective dictionaries. Entering TURNAROUND into the noun dictionary prevents the false verb from being triggered, and the result is that lead is not coded under TURN.[14] But there is no way that anyone could have predicted such a misapplication of the verb phrase when TURN IN GUN was entered in the verb dictionary. It was a pretty good entry.[15]

Deverbing, then, is a crucial task of the coder. Watching for English's random words like "turnaround" is the easy way of eliminating false verbs, though. Life gets more complicated with systematic sources of false verbs, like the ending "ing." Here again, English's use of identical words in multiple capacities creates quandaries. Often, -ing

---

[14] It will probably end up being coded as ISRAEL SENT TANKS INTO GAZA_STRIP AND THE_WEST_BACK. This is not correct, strictly speaking, because the lead is reporting an "apparent turnaround" instead of an attack. On the other hand, the attack did occur at some point, so coding an attack is probably more accurate than what was coded at first. One could argue that Sharon's "apparent turnaround" from the use of military force must have been a form of accommodation, in which case the lead had the right code for the wrong reason in the first case. Even then, though, it would be best to make the noun dictionary addition. Erroneous codes that happen to be right are not systematic; it is preferable to have the better-developed dictionary and one semi-correct lead than to leave the dictionary unaltered—causing who knows what problems in future leads—and get lucky in this one case. And you thought coding looked easy at first glance…

[15] "Pretty good" because there is one problem with the phrase. The word IN should have been followed by an underscore: IN_. This would have prevented "into" from triggering it, though it would not have mattered in this particular lead because there is another "in" later on in the sentence. The bigger picture, though, is that prepositions, conjunctions, and any word that cannot be stemmed in English should be followed by a terminal underscore when entered in the verb dictionary. This prevents "intestine," "insoluble," and "intergalactic" from being recognized as "in." The most common (though certainly not the only) words this should be used for include "in," "on," "of," "with," "no," "up," and the ubiquitous "to."

forms of verbs are used as nouns and are irrelevant to the coding of a lead: "Egypt's president discussed the fighting in the West Bank with Lebanon's prime minister." TABARI will code "fighting" as a stem off the verb FIGHT, even though it does not act as a verb in the sentence. This is not a problem in this particular lead. However, it becomes problematic if the lead is longer:

> Egypt's president DISCUSSED the FIGHTING in the West Bank with Lebanon's prime minister when he LANDED in Beirut after he VISITED Turkey to SIGN a cooperation accord regarding freight SHIPPED though Suez to Istanbul.

Now the sentence has six words recognized as verbs; it will not be coded. Entering "fighting" into the noun dictionary will prevent it from being recognized as a verb and allow the lead to generate the events

EGY	LEB	020	(Consult)	EGYPT DISCUSSED WITH LEBANON'S
LEB	EGY	020	(Consult)	EGYPT DISCUSSED WITH LEBANON'S

On the surface it would appear that all words ending in –ing could be coded as nouns. However, this would cause other problems. Often, it is useful to have –ing forms recognized as verbs; the classic example of this is "meeting." Leads commonly take the form "Egypt's president is meeting with Lebanon's prime minister today." Here there is no other verb to grab on to other than "meeting." It turns out that while occasionally "meeting" causes trouble by being coded as a verb, more often it is beneficial to call it a verb, especially as meetings make up such a large proportion of the events in international diplomacy. Rather than being a false verb, then, "meeting" is a "useful (pseudo-)verb." The example given above with "fighting" is not the best, either, because the verb "fighting" might be useful as a trigger for the CAMEO code 190, the use of conventional military force. On the other hand, it might not. The question is whether more leads are coded correctly when "fighting" is left as a verb or when it is considered a noun.

The trick, of course, is to know which –ing words are false verbs and which are useful (pseudo-)verbs. This is a matter of experience and the writing style of the text being coded. All a coder can do is be sensitive to the trends in the text and accommodate them with verb and noun dictionary entries. Once again the issue is a balancing act where the coder is constantly seeking the approach that will maximize useful codes and minimize miscodes and rejected leads.

Creative phrasing can also reduce the number of false verbs in a sentence. For example, "arm," "force," "free," and "trade" are all words that might usefully be included in the verbs dictionary. However, entering "armed_forces" and "free_trade_zone" as nouns will prevent these sequences of words not used as verbs from being recognized as multiple verbs. In one fell swoop several problems have been eliminated.

<u>The End</u>

Specific approaches to specific scenarios are a useful starting point, but no finite list can ever address every situation the coder will confront. Hopefully these suggestions have given a sketch of the mindset and approach that might allow a coder to quickly and

effectively defeat new problems as they arise. This mindset boils down to a few basic points:

Balance: Walk the line between opposite extremes.

Perspective: Always keep the big picture in mind. Don't give undue attention to single eccentric leads. Don't spend too much effort on internal political events within a single country if the project is aimed at international relations. Remember what the project wants to discover and concentrate on extracting it from amidst the extraneous information.

Balance and Perspective: When making dictionary changes, consider what effect they will have when applied over thousands of leads. Don't be too specific or too general in wording and codes. Don't make large changes without consideration of the consequences, both intended and unintended.

Creativity: Solve problems in flexible ways, keeping in mind the above considerations. Be willing to use one part of speech as another, if it will systematically work to solve a problem.

Happy coding!

Miscellaneous Thoughts

Here are some suggestions, in no apparent order:

Web searching is a pretty inefficient way of doing date restrictions, though google will usually get the job done if you look long enough. Searching AFP on Lexis-Nexis is better, especially for the purposes of the KEDS project, because this will turn up the dates and titles that are applicable to the project—the ones that will actually appear in the data. This, though, is a pretty incomplete picture. To get more information, try these websites:

www.rulers.org[16]

http://www.terra.es/personal2/monolith/00index.htm

The second site above is more detailed than the first as far as listing cabinet officers, etc, but it does not go back as far. It also contains fairly long biographies of many political leaders that could be invaluable for building date restrictions; the problem is that they are in Spanish. Auto-translating using google wasn't particularly helpful, either. Rulers.org at one time had brief bios but then took them down; it appears that they intend to repost them at some point, but it has not happened yet.

Working alone allows much higher productivity. This is especially noticeable when working in the evening, away from the assorted noises of the office environment. There are some pretty good views of Lawrence in the dark from the windows, too.

[16] A fantastic site; fascinating even outside of the context of needing date restrictions

Know the coding scheme. Spending time learning it allows for much faster coding and saves time in the long run.

Don't try to code for too long at a time. No one can do it well. Two hours is about the limit for most people in one sitting. Even making it that long is sometimes a challenge.

Parse, parse, parse. It's not quite magic, but it's close. And know what the parse output symbols mean. Parsing quickly reveals which words the program is reading and which are being ignored; how each word is being treated, and how many verbs there are. It doesn't yet highlight which words are being matched to a pattern in the dictionary, but maybe some day…

Use terminal underscores on prepositions, etc. If you don't want to force two words to be next to each other, use an underscore and then a space.

Watch out for stemming and words with multiple meanings. These are the sources of a large portion of miscodes.

"Meet" codes are almost always reciprocal. Don't forget the second half of 022:023, 020:020, 021:021, and 026:026. Also, "Agree": 060:060, etc.

Discard codes are fun and profitable. Just don't use them lightly.

Be systematic when assigning actor codes. For substate actors, use the first three letters to indicate geographical area and then get progressively smaller in scope with each subsequent three-letter suffix. BOSSER is people of Serbian ethnicity in Bosnia. CROSER is people of Serbian ethnicity in Croatia. BOSSERMOS is Muslim people of Serbian ethnicity in Bosnia.

<u>For Phil and Misty's eyes only:</u>
Not really. These just won't be helpful for coders. You probably know most of them already…

At some point I think it would be useful to comb through the dictionaries to look for redundancies (for some reason some entries are identical duplicates), errors (bad codes, useless phrases), and things that need to be removed or consolidated (some verb dictionary main headings from the early 90s (wow, I can't believe I get to say "the early nineties") probably need to be reconsidered in light of the recent upgrades to the program, etc.)

Consistency is crucial when assigning event codes to phrases; this requires good communication between individual coders and between project directors and coders. This is the most important during the initial training phase.

A cooler room temperature helps prevent coders from falling asleep.

Having other projects to break up the coding is a good idea. Researching date restrictions, downloading new data, etc. give coders a chance to use different parts of their brains.