Excel Extractor: A Worksheet Tool for Extracting Information from Case Narratives in the Violent Death Reporting System

Elise Omaki[1]; Felipe Restrepo[2]; Wendy Shields[1]; Alan Abrahams[2]

[1] Johns Hopkins Center for Injury Research & Policy, Johns Hopkins Bloomberg School of Public Health, Baltimore, MD

[2] Pamplin College of Business, Virginia Polytechnic Institute, Blacksburg, VA

1. Program Description

The Excel Extractor is a customized natural-text information extraction tool created in Microsoft Excel. To guarantee ease-of-use and platform compatibility, the Excel Extractor was created exclusively with native Excel functions (such as MID, FIND, and SUBSTITUTE). It has no macros, VBA scripts, or connections to external programs, i.e., software security concerns are allayed as there is no custom imperative programming language code, and it will work on any offline machine, as long as the machine has base Microsoft Excel installed. The Excel Extractor follows three steps to extract data. First, an indicator dictionary is used to identify the narratives' relevant sections. Next, a keyword dictionary is used to extract the desired information from the relevant sections. Lastly, extracted information is then mapped to label values. Complete versions of the spreadsheet tool are available at: https://github.com/omakielise/Excel-Extractor.

Step 1: Generate the indicator-term array.

The indicator dictionary is used to identify relevant sections of the narrative. It lists terms that 'lead up to' or are frequently used within the context of the desired information. For example, when extracting who found the victim, indicator terms include "by", "his" and "her" (see Figure 1).

Q4_indicator	instance
by	1
by	2
his	1
her	1

Figure 1 - Sample indicator dictionary

Because native Excel formulas do not have an integrated way to loop through all instances of the indicator term within a narrative, an 'instance' counter is used by adding multiple instances of the term to the indicator dictionary. For example, if the user wants to extract narrative sections for both the first and second time the term "by" appears in the narrative, it is added twice to the indicator dictionary (see Figure 1).

The relevant sections of the narrative are then extracted using the following formula:

```
=TRANSPOSE(IFERROR(MID($E2,FIND("~",SUBSTITUTE($E2,$A$2:$A$5,"~",$B$2:$B$5
))-80,80),"")
&IFERROR(MID($E2,FIND("~",SUBSTITUTE($E2,$A$2:$A$5,"~",$B$2:$B$5)),160),""))
```

where, *$E2* refers to the text narrative of interest, *$A$2:$A$6* refers to the terms in the indicator dictionary, and *$B$2:$B$6* refers to the desired instance for each of the terms. This example is a two-part formula that extracts text beginning 80 characters before the indicator term and 160 characters after the indicator term, concatenating the extracted sections via '*&*'.

Both parts of the formula work in the same way. The MID() function extracts a specific number of characters starting at a specified character number from a given text (i.e. the narrative). In the

above MID() function, the character number to begin extracting text from is determined by the FIND() function, which returns the character number location of a specified term.

We use the SUBSTITUTE() function, which allows the user to replace any instance of a term with a specified string, to loop through multiple instances of the same term within the narrative. By substituting the indicator term with "~", FIND() will return the character number location of the indicator term instance.

Thus, this nested FIND() SUBSTITUTE() function returns the MID() function's starting character number parameter, i.e. where the indicator term will begin its extraction. This function, wrapped in TRANSPOSE(), so that it is displayed horizontally, returns an array size equal to the length of the indicator dictionary, with one array entry, per cell, for each dictionary entry, as seen in Figure 2. Indicator terms that are found in the narrative will return the corresponding text section, while non-matches will return a blank cell.

ID	Q4_indicator-term1	Q4_indicator-term2	Q4_indicator-term3	Q4_indicator-term4
1	by her husband who last knew h	o last knew her to be alive approxima	ted by EMS to the hospital where she	her residence by her husband who last
2	by his father who last new V aliv		his apartment by his father who last n	her who last new V alive 8 hours prior.
3	by her friend who was the last p		r to the incident. EMS responded and	her residence by her friend who was tl

Figure 2 - Sample indicator-term array

Step 2: Creating the keyword-term array.

In this step, the keyword terms of interest are extracted from the indicator-term array that was generated in Step 1, using a keyword dictionary. In our example extracting who found the victim, keyword terms may include "girlfriend", "mother", "roommate" and other relations known to find a decedent, as well as variations of those terms. For each attribute of interest, terms in the keyword dictionary can be generated by packages such as Fumeus[1], and/or manually curated by content matter experts.

The following formula is used to perform the keyword term extraction from the indicator-term array:

=MID(G2#,MIN(IFERROR(SEARCH($A$9:$A$16,G2#),"")),MAX(LEN($A$9:$A$16)) + 20)

where *G2#* refers to the array extracted from the narrative in Part 1 and *$A$9:$A$16* refers to the keyword dictionary. Similar to Step 1, the MID function is used to extract a keyword term. However, in this case, the text is found in the form of an array, thus, the formula is applied to every array entry. Hence, the MID() function text parameter is the array, and the extraction starting character number is determined by the SEARCH() function.

The SEARCH() function looks for the terms in the keyword dictionary within each cell in the indicator-term array, returning an array of length equal to the keyword dictionary and width equal to that of the indicator-term array. Keyword matches have their character number location returned within the array, while non-matches return a blank cell (via the IFERROR() function). The MIN() function is then used to reduce the array yielded to a single value – the earliest location of a keyword match found within the indicator-term array.

[1] Goldberg DM, Gruss RJ, Abrahams AS. Fumeus: A family of Python tools for text mining with smoke terms. Software Impacts. 2022;12. doi:10.1016/j.simpa.2022.100270

The MID() function's "specific number of characters to extract" parameter is determined by taking the MAX() of the length (LEN()) of all the terms in the key-term dictionary and adding 20 characters to ensure that there are no partial key-term extractions. Hence, the above function will return a keyword array the same size as the indicator array, but now only containing the keywords plus a few surrounding words (see Figure 3). Note that since the earliest keyword match is used for all instances within each indicator-term array, certain entries in the keyword-term array will not display a matching keyword.

ID	Q4_keyword-term1	Q4_keyword-term2	Q4_keyword-term3	Q4_keyword-term4
1	husband who last knew her to b	knew her to be alive approxima	EMS to the hospital where she	idence by her husband who last
2	father who last new V alive 8		rtment by his father who last	last new V alive 8 hours prio
3	friend who was the last person		e incident. EMS responded and	idence by her friend who was t

Figure 3 - Sample keyword-term array

The keyword-term array is further reduced using the following formula:

=TRIM(TEXTJOIN(",",TRUE,IFERROR(MID(G10#,SEARCH($A$9:$A$16,G10#),13),"")))

where *G10#* refers to the keyword-term array and *$A$9:$A$16* refers to the keyword dictionary. The MID() SEARCH() nested formula is again used to locate the key-term's location within the array, this time extracting only 13 characters from the starting location of the keyword. In the case the keyword-term array does not have a keyword match, the IFERROR() function is employed to return a blank cell.

The TEXTJOIN() function is used to condense the resulting array, joining all keyword-term array entries together using a comma separator. Finally, the TRIM() function is wrapped around the entire formula to eliminate extra whitespaces (see Figure 5).

Step 3: Translating the condensed keyword-term array to class labels.

At this point, depending on user needs, the condensed keyword-term array can be either converted to binary or multi-class.

If converting to binary, a simple formula is used:

=IF(LEN(G18)>0,1,0)

indicating that if the cells in the condensed keyword-term array have any text, i.e., there was a keyword match within the desired context (indicator-term array), the class label is 1, and 0 otherwise.

If the conversion is to multi-class, a longer and more complex IF() function is used to assign the condensed keyword term array text to a class label. Every entry within the keyword dictionary is assigned to a specific class label using a combination of IF(), SEARCH(), and MAX() formulas. In our example extracting who found the victim, the following (abbreviated) formula is used to assign the collapsed keyword-term array text to its pertaining classes:

```
=IF(MAX(IFERROR(SEARCH({"mother","father"},G18),0))>0,"PARENT",
IF(MAX(IFERROR(SEARCH({"boyfriend","girlfriend","wife","husband"},G18),0))>0,"SIG
OTHER",IF(MAX(IFERROR(SEARCH({"friend","colleague"},G18),0))>0,"FRIEND",
IF(MAX(IFERROR(SEARCH({"unknown","whom"},G18),0))>0,"UNK",
IF(LEN(G18)=0,"UNK","OTHER")))))
```

Entries in the keyword dictionary pertaining to the same class are searched for, within the collapsed keyword-term array in the same SEARCH() instantiation, via listing them inside {}.

Since an array size equal to the number of terms in {} is returned, the MAX() function is used to collapse this array and return the largest value yielded by the SEARCH() function.

2. Comment

While the Excel Extractor was purposely designed to extract specific information about drug overdose deaths from case narratives in the Maryland Violent Death Reporting System (VDRS) data, its adaptation and application has far-reaching potential for public health surveillance. The flexible design allows for tailoring the indicator and keyword dictionaries based on the information of interest, and formulae can be modified to extract and reclassify labels as needed. As such, the tool could be used on VDRS data from other states, expanded to identify information of interest regarding other mechanisms of injury and violence, or adapted to similar data sources such as the State Unintentional Drug Overdose Reporting System (SUDORS). The program has utility for prospective use by reducing the lag for producing and publishing data that researchers and practitioners can act on, and it can be used retrospectively, for example by establishing rules for the Excel Extractor consistent with newly added variables and applying it to historic data.

While this program has promise, it should be noted that the Excel Extractor was designed for use on the condensed narratives contained in the VDRS database. These narratives are prepared by trained data abstractors following a review of lengthy reports from the medical examiner and law enforcement. Additional research is needed to identify if a similar approach can be used on the raw narratives.

3. Acknowledgements

This work was funded by the American Public Health Association’s Data Science Demonstration Project, and through a Discovery Award from Johns Hopkins University.