Analyzing Survey Text

Jon Peck

jkpeck@gmail.com

16 April, 2021
Updated 27 February, 2022

It is common for surveys to include open ended questions. Such questions come in two main types. The first is an "other" category in a question that also has a list of specified choices. Examples are job title, medical conditions, race, political affiliation, and brand of car owned. The second is an attitude or opinion question or a narrative. Examples include opinion about a company, political candidate, or issue. There might be a narrative of a customer service problem or an explanation of why a rating was assigned in a previous question.

Questions with unstructured text answers can contain valuable information, but they are difficult to analyze, so the survey users may be reduced to just reading through the answers. Text might be coded into structured form, but coding is time consuming and expensive, and it can be very subjective. The coding may be inconsistent from respondent to respondent, especially if there are multiple coders.

Nevertheless, there are ways to extract the information in such questions, and combining this with the more structured information can improve the analysis. Respondents may well answer open ended questions differently from structured equivalents, so it can be important to extract this information even if it has a structured counterpart. Also, a pilot study may use open ended questions, and the information in the responses used to construct a more structured survey.

The STATS TEXTANALYSIS extension command for SPSS Statistics provides tools for extracting the information in unstructured responses that can then be combined with the structured data. The command provides tools for preprocessing the data and setting the text extraction parameters, for calculating frequencies and sentiment scores, and for constructing dummy variables based on the content. For explanations of the operational details, use the dialog box or syntax help.

Installation of this extension requires some extra steps. See the Installation section below for details.

Analysis

We will assume that the preliminary steps have already been taken and look at the statistical analysis first and then return to discuss the preliminary issues.

We will use data from the American National Elections Study 2008 for these examples (https://electionstudies.org/data-center/2008-time-series-study/). The study has a number of dimensions that we will ignore for this exposition.

Frequencies

The most basic analysis is to count word and phrase frequencies in a question.   Finding the most frequent of these gives an overview of the responses. It can also help in setting up coding instructions

or in devising recodes where *Other* categories should be mapped into the structured responses. The frequency tabulations intelligently identify words, ignoring the letter case. First, we tabulate the most frequent words in an open-ended artificial dataset and then we will look at likes for the Democratic candidate in the presidential election. Frequencies produces three tables – individual words, bigrams (two words together), and trigrams (three words together). Of the approximately 2300 cases, almost half have text in this question, but often text questions are much sparser.

If the dataset is weighted, the weights are used in the Frequency tables, but they are rounded to integers. This means in particular that weights less than .5 will cause the case to be omitted from the table. Although the standard Statistics weight is a replication weight, fractional weights are sometimes used. If you need more precision in the counts, you can multiply the weight by, say, 10 so that rounding distorts the results less, but don't use the multiplied weights for analysis in other procedures. Bear in mind that weights are implemented by replicating the words in the data set, so very large weights may cause problems.

The definition of n-grams is more complicated than it might seem at first sight. In this tool, the process is as follows.

- For each case, find the sentences in the variable text
- For each sentence, find the words it contains and convert to lower case
- Stem, if specified, and remove stopwords
- Count occurrences of words, adjacent pairs, and adjacent triples in each case
- Sum across the sentences in the case

Duplicates in the sequence are discarded, so "a", "b", "b" or "a", "b", "a" would not be counted as trigrams. By processing each sentence separately, an n-gram is prevented from spanning sentences.

Here is a very simple dataset with the frequency output (footnotes omitted). The data are weighted by w.

	w	x
1	1	aa bb cc dd
2	1	aa aa bb bb
3	2	bb cc dd dd
4	2	aa. bb aa

Word Frequencies for x a,b,c,d,e

	Word Frequency
aa	7
bb	7
dd	5
cc	3

Bigram Frequencies for x a,b,c,d,e

	Bigram Frequency
bb cc	3
cc dd	3
aa bb	2
bb aa	2

Trigram Frequencies for x a,b,c,d,e

	Trigram Frequency
bb cc dd	3
aa bb cc	1

For the real dataset mentioned above, this is the output, The dataset has fractional weights. The footnote counting cases with text is unweighted.

Word Frequencies
for A8bDemPC_like

A8b. DemPC_like

a,b,c,

	Word Frequency
change	260
like	243
people	192
good	140
think	132
seems	93
health	91
class	87
would	83
democrat	80

a. Case and stopwords are ignored

b. Words have not been stemmed

c. 10 most common items

d. 1495 cases have text

e. Frequencies are based on rounded weights

The bigram and trigram cases for this question are more interesting. This would be harder to see without this tool.

Bigram Frequencies for A8bDemPC_like

A8b. DemPC_like[a,b,c,d,e]

	Bigram Frequency
middle class	67
health care	59
need change	34
seems like	22
new ideas	19
good speaker	18
poor people	17
tax breaks	16
good president	15
health insurance	15

a. Case and stopwords are ignored

b. Words have not been stemmed

c. 10 most common items

d. 1495 cases have text

e. Frequencies are based on rounded weights

Trigram Frequencies for A8bDemPC_like A8b. DemPC_like [a,b,c,d,e]	Trigram Frequency
tax breaks middle	9
breaks middle class	8
make good president	8
help middle class	7
middle class people	7
first black president	6
lesser two evils	6
bring people together	5
bring troops home	5
like change sounds	5

a. Case and stopwords are ignored

b. Words have not been stemmed

c. 10 most common items

d. 1495 cases have text

e. Frequencies are based on rounded weights

The footnotes on the tables report that stopwords are ignored, and there is no stemming. Stopwords are frequently occurring but mostly nonmeaningful words that should generally be ignored. Stopwords are, of course, language specific. The tool supports many languages. The English stopwords are these: didn, won, then, here, are, being, been, other, both, this, should've, hasn, mustn't, whom, shan't, wasn, so, t, haven, your, did, not, shan, aren't, and, while, them, needn, you'll, no, same, she, to, few, doesn, itself, ma, wasn't, wouldn, will, with, above, ve, yourselves, it, themselves, until, out, now, all, him, own, he, because, can, their, up, down, again, than, shouldn, below, didn't, our, how, isn't, his, by, on, i, couldn, me, hadn, ourselves, they, does, during, just, hadn't, isn, that'll, once, yours, why, into, that, won't, her, you, more, aren, further, most, or, through, you'd, haven't, those, ain, very, re, hasn't, you've, had, has, do, after, don't, each, about, doesn't, m, y, having, too, am, is, there, any, don, what, herself, weren't, wouldn't, when, as, weren, we, were, myself, a, mustn, in, mightn't, it's, have, himself, should, before, off, some, against, over, couldn't, shouldn't, s, be, my, between, mightn, theirs, these, o,

who, which, hers, of, where, ours, but, for, you're, was, only, doing, needn't, nor, its, if, from, ll, she's, yourself, d, the, an, under, at, such.

Frequencies always ignore them.

Stemming means removing morphological affixes from words. That is, reducing a word to its root or stem, which might not even be a word. This, too, is language specific. The tool supports many languages. The results can be surprising. For example, *meetings* stems not to *meeting* but all the way to *meet*. Especially for word searching, it is important to know the stems. Therefore, the tool provides for creating a variable that shows the stemmed words for each case.

Sentiment

Frequencies are useful for both *Other* situations and opinions and narratives. Sentiment analysis applies mainly to the latter. Sentiment analysis is used to determine the degree to which text is negative, neutral, positive, or a composite of those. The analyzer used here is described in

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

It is best suited for small amounts of text as might be expressed in Twitter and similar social media.

The tool calculates one to four measures of the sentiment, Vader scores, of a variable for each case using a text lexicon that assigns a score to each word along with intensifiers such as greatly, barely, and other such words. It even accounts for some phrases such as *cut the mustard*. It takes into account negatives such as *not*, *doesn't*, and *ain't*. A complete list of these items can be obtaining using the Utilities tab on the dialog box.

The negative, neutral, and positive measures sum to 1. The compound measure is a heuristic that is not just the average of the other measures.

This table gives an idea of how these scores perform.

text	text_neg	text_neu	text_pos	text_comp
The pie was terrible	0.51	0.49	0.00	-0.48
The pie was not good	0.38	0.62	0.00	-0.34
The service was soso	0.00	1.00	0.00	0.00
The pie was okay	0.00	0.61	0.39	0.23
The pie was good	0.00	0.51	0.49	0.44
The pie could have been better	0.00	0.63	0.37	0.44
The pie was very good	0.00	0.56	0.44	0.49
The pie was very, very good	0.00	0.59	0.41	0.54
The pie was great	0.00	0.42	0.58	0.62

The data have been sorted by the overall score, text_comp. Unknown words are considered neutral. Negative compound scores are negative sentiment, and positive are positive sentiment. Notice that not

only did *very* increase the rating of good, but *very, very* increased it even more. *Not good* was recognized as a negative.

Sentiments use a lexicon that can be modified. The tool can create a dataset of the lexicon scores and words and datasets of negation and emphasis terms. You can add or change words and scores based on that lexicon, including negation and emphasis terms. However, the analysis is based on English grammar. It has been suggested that machine translation of the text before calculating sentiment may be a viable option, but your mileage may vary.

Spelling Correction

To level the playing field (an idiom that Sentiment does not understand), the tool includes a spelling corrector. Corrections are based on Levenshtein distance, which is the number of changes required to change a word into a recognized word. The choice is then weighted by word frequency. You can specify that recognized names not be corrected. The checker has over 7500 names built in. The "corrected" text is written to a new variable.

The checker is not interactive and may well be wrong, so experimentation may be necessary in determining whether to use it or not. Alternatively, use the interactive spell check tool built into Statistics. You can improve spelling correction performance by creating a dataset with only the variables for which you want to correct spelling and then merging the result back with the main dataset.

Some foreign languages are supported, but you can add your own spelling dictionary. If correcting data with a specialized vocabulary, which might be jargon, organization terms, abbreviations, or other nonstandard words, results will improve if you supply a supplemental dictionary.

Searching

Identifying specific words in text variables, perhaps based on the frequency analysis, can be useful in further analysis of the cases, including clustering and segmentation modeling. The tool includes a method for searching variables for word and n-gram occurrences. It creates a dummy variable for the presence of any or all or a pattern for a set of specified terms with or without stemming. Using pattern, the result is a string variable with a sequence of 1 and 0 values with one value per search word or n-gram in the order listed. The output variable for cases with no text in a variable will have the system missing value if numeric (any or all) and will, therefore, be excluded from statistical procedures. For pattern, the result will be blank, and blank is declared as a missing value.

To specify a bigram or trigram, enter a dash between the items in the search list, e.g., good-time would be a bigram.

Here is a frequency table for a pattern search variable with three words.

A8bDemPC_like_ser					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	000	1126	48.5	75.3	75.3
	001	73	3.1	4.9	80.2
	010	13	0.6	0.9	81.1
	011	3	0.1	0.2	81.3
	100	266	11.5	17.8	99.1
	101	13	0.6	0.9	99.9
	110	1	0.0	0.1	100.0
	Total	1495	64.4	100.0	
Missing		827	35.6		
Total		2322	100.0		

It shows that 1126 cases had text but no occurrences of any of the search words; 73 cases had only the last word, 13 cases had only the second word, 3 cases had words two and three, and so on.

Synonyms Besides searching for exact terms, you can include synonyms in the search. If any synonym is found, that counts as a hit for the term. You can filter by part of speech or suppress synonym search.

Symbol	Definition
x	do not search for synonyms
n	noun
v	verb
a	adjective
r	adverb
y	any

For bigrams, specify two symbols and for trigrams specify three. For example, xn would mean no synonyms for the first word in a bigram – just the word itself - followed by a noun synonym.

For n-grams, each synonym for the first word is combined with each synonym for the second word and so on, so the possibilities can get quite large. Consider using x for some words in an n-gram.

Check the **Display synonyms** box to see the sets of synonyms. Here is an example. You can see three different meanings – calculate, judge, and think.

Search Synonyms[a]

	Word, Bigram, and Trigram Synonyms

reckonings(y)	figure, see, opine, cypher, calculate, work-out, imagine, reckonings, consider, computation, calculation, think, enumeration, estimate, cipher, reckoning, count, compute, forecast, regard, counting, bet, look, view, numeration, figuring, count-on, reckon, suppose, depend, tally, guess

a. Parenthesized synonyms are bigrams or trigrams

Here is an example for the trigram he-is-young with part of speech specified as xxn, the synonym set is

(he, is, Whitney-Young), (he, is, Lester-Willis-Young), (he, is, Pres-Young), (he, is, Cy-Young), (he, is, Brigham-Young), (he, is, young), (he, is, Danton-True-Young), (he, is, new), (he, is, offspring), (he, is, Loretta-Young), (he, is, unseasoned), (he, is, youthful), (he, is, Whitney-Moore-Young-Jr.), (he, is, immature), (he, is, vernal), (he, is, Edward-Young), (he, is, youth), (he, is, untried), (he, is, Thomas-Young), (he, is, untested), (he, is, Young)

Any of these items would be considered a hit. Notice that besides several proper names, synonyms include young, youthful, immature, vernal, youth, untried, and untested following he is. Synonyms can sometimes produce unexpected results. For example, synonyms for "he" include "helium" and "atomic-number-2".

> **Commented [JP1]:** i

The term itself is always included in the synonym set, so an unknown word will have itself as a synonym.

By default, synonym search uses English, but a number of foreign languages can be specified.

A custom variable attribute named search is created that records the search criterion. From patterns, you can conveniently create a set of dummy variables using the SPSSINC CREATE DUMMIES extension command. The pattern or dummy variables might be used in model building with TWOSTEP CLUSTER, REGRESSION, or many other procedures.

Installation with SPSS Statistics Version 27

This procedure requires several additional items. After installing it, do the following. Depending on your system setup, you might need to do these steps in Administrator mode. Install the STATS PACKAGE INSTALL extension command, which will appear on the Extensions menu, and use it to install nltk and pyspellchecker

- Open a syntax window in Statistics, and run this code.
  BEGIN PROGRAM PYTHON3.
  ```
  import nltk
  nltk.download()
  ```
  END PROGRAM.
  This will display a window with a table of items you can add to your installation. Select at least names, stopwords, wordnet, and vader_lexicon.
- Optionally, go to spelling dictionary as mentioned above and extract the words.txt file from words.zip. Specify that location when you run the procedure.

- Install the SPSSINC TRANS extension command via the Statistics Exensions > Extension Hub menu.

Installation with SPSS Statistics Version 28 and Later

This procedure requires several additional items. After installing it, do the following. Depending on your system setup, you might need to do these steps in Administrator mode. It is no longer necessary to install a separate Python distribution. Most of the additional modules used by this procedure will be installed with it, but some additional steps are necessary.

- Start Statistics. You might need to run it as Administrator depending on your security settings.
- Run this code from the Syntax Editor window.
  ```
  begin program python3.
  nltk.download()
  end program.
  ```
  This will bring up a window listing the packages available for nltk. Click on *All Packages* and choose at least names, stopwords, wordnet, and vader_lexicon.
- Optionally, go to spelling dictionary and extract the words.txt file from words.zip. Specify that location when you run the procedure.
- Install the SPSSINC TRANS extension command via the Statistics Extensions > Extension Hub menu.

Acknowledgements

STATS TEXTANALYSIS relies on open source tools for language processing and spelling correction.

Language processing:
- The NLTK project is led by Steven Bird and Liling Tan. Individual packages are maintained by the following people:
- Semantics

    - Dan Garrette, Austin, USA (nltk.sem, nltk.inference)
- Parsing

    - Peter Ljunglöf, Gothenburg, Sweden (nltk.parse, nltk.featstruct)
- Metrics

    - Joel Nothman, Sydney, Australia (nltk.metrics, nltk.tokenize.punkt)
    - Python 3

    - Mikhail Korobov, Ekaterinburg, Russia
        - Releases

    - Steven Bird, Melbourne, Australia
        - NLTK-Users

- Alexis Dimitriadis, Utrecht, Netherlands

- Wordnet

o George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

- *WordNet: An Electronic Lexical Database*(link is external) (citation above) is available from MIT Press(link is external).

- Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.
- Citing WordNet | WordNet (princeton.edu)
- Wordnet license: License and Commercial Use of WordNet | WordNet (princeton.edu)

Spelling:
- Author: Tyler Barrus
- Peter Norvig blog post on setting up a simple spell checking algorithm
- P Lison and J Tiedemann, 2016, OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)