# A practical primer on processing semantic property norm data

Erin M. Buchanan[1], Simon De Deyne[2], & Maria Montefinese[3]

[1] Harrisburg University of Science and Technology

[2] The University of Adelaide

[3] University of Padua

Author Note

Any suggested author note?

Correspondence concerning this article should be addressed to Erin M. Buchanan, 326 Market St., Harrisburg, PA 17101. E-mail: ebuchanan@harrisburgu.edu

## Abstract

Semantic property listing tasks require participants to generate short propositions (e.g., <barks>, <has fur>) for a specific concept (e.g., dog). This task is the cornerstone of the creation of semantic property norms which are essential for modelling, stimuli creation, and understanding similarity between concepts. However, despite the wide applicability of semantic property norms for a large variety of concepts across different groups of people, the methodological aspects of the property listing task have received less attention, even though the procedure and processing of the data can substantially affect the nature and quality of the measures derived from them. The goal of this paper is to provide a practical primer on how to collect and process semantic property norms. We will discuss the key methods to elicit semantic properties and compare different methods to derive meaningful representations from them. This will cover the role of instructions and test context, property pre-processing (e.g., lemmatization), property weighting, and relationship encoding using ontologies. With these choices in mind, we propose and demonstrate a processing pipeline that transparently documents these steps resulting in improved comparability across different studies. The impact of these choices will be demonstrated using intrinsic (e.g. reliability, number of properties) and extrinsic measures (e.g., categorization, semantic similarity, lexical processing). Example data and the impact of choice decisions will be provided. This practical primer will offer potential solutions to several longstanding problems and allow researchers to develop new property listing norms overcoming the constraints of previous studies.


*Keywords:* semantic, property norm task, tutorial

# A practical primer on processing semantic property norm data

1. Available feature norms and their format

- Property listing task original work: Toglia and Battig (1978); Toglia (2009); Rosch and Mervis (1975); Ashcraft (1978)
- English: McRae, Cree, Seidenberg, and McNorgan (2005), Vinson and Vigliocco (2008), Buchanan, Holmes, Teasley, and Hutchison (2013), Devereux, Tyler, Geertzen, and Randall (2014), Buchanan, Valentine, and Maxwell (2019)
- Italian: Montefinese, Ambrosini, Fairfield, and Mammarella (2013); Reverberi, Capitani, and Laiacona (2004), Kremer and Baroni (2011)
- German: Kremer and Baroni (2011)
- Portuguese: Stein and de Azevedo Gomes (2009)
- Spanish: Vivas, Vivas, Comesaña, Coni, and Vorano (2017)
- Dutch: Ruts et al. (2004)
- Blind participants: Lenci, Baroni, Cazzolli, and Marotta (2013)

I'm sure there are more, here's what we cited recently.

Define concept, feature for clarity throughout - make sure you use these two terms consistently.

2. Pointers about how to collect the data

a. instructions, generation, verification, importance

I really like the way the CSLB did it: https://cslb.psychol.cam.ac.uk/propnorms

They showed the concept, then had a drop down menu for is/has/does, and then the participant typed in a final window. That type of system would solve about half the problems I am going to describe below about using multi-word sequences. Might be some other suggestions, but for that type of processing, you could do combinations and have more consistent data easily.

3. Typical operations performed on features

Due to the productivity in language, a semantic feature can be expressed in a myriad of ways. Without any further processing, many features will be expressed in an idiosyncratic way, despite the fact that they capture the same meaning. For example, the fact that bicycles have two wheels is expressed as „ , , . The next sections provide a tutorial on how data from the semantic feature listing (SFL) task might be processed from raw input to a more compact feature output. The tutorial is written for R and is fully documented, such that users can adapt it to their language of choice. Figure 1 portrays the proposed set of steps including spell checking, lemmatization, exclusion of stop words, and final processing in a multi-word sequence approach or a bag of words approach. After detailing these steps, the final data form will compared to previous norms to determine the usefulness of this approach.

## Materials and Data Format

The data for this tutorial includes 16544 unique concept-feature responses for 226 concepts from Buchanan et al. (2019). The concepts were taken from McRae et al. (2005), Vinson and Vigliocco (2008), and Bruni, Tran, and Baroni (2014). The concepts include 185 nouns, 25 verbs, and 16 adjectives. Concreteness ratings collected by (**???**) were matched with the current data set. The concreteness ratings can range from 1 (*abstract (language based)*) to 5 (*concrete (experience based)*). The nouns were rated as most concrete: $M = 4.59$ ($SD = 0.52$), followed by adjectives: $M = 3.78$ ($SD = 0.81$), and verbs: $M = 3.57$ ($SD = 0.79$). The data consist of a text file where concept-feature observation is a row and each column is a variable. An example of this raw data is shown in Table 1. The original data can be found at https://osf.io/cjyzw/.

This data was collected using the instructions provided by McRae et al. (2005), however, in contrast to the suggestions for consistency detailed above (Devereux et al., 2014), each participant was simply given a large text box to include their answer. Each answer includes multiple embedded features, and the tutorial proceeds to demonstrate potential processing addressing the data in this nature. With structured data entry for participants, the suggested processing steps are reduced.

## Spelling

Spell checking can be automated with the `hunspell` package in *R* (Ooms, 2018). Each `answer` can be checked for misspellings across an entire column of answers, which is located in the `master` dataset. The default dictionary is American English, and the `hunspell` vignettes provide details on how to import your own dictionary for non-English languages. The choice of dictionary should also normalize between multiple varieties of the same language, for example, the `"en_GB"` would convert to British English spellings.

```
## Lower case to normalize
master$answer <- tolower(master$answer)
## Install the hunspell package if necessary
#install.packages("hunspell")
library(hunspell)
## Check the participant answers
## The output is a list of spelling errors for each line
spelling_errors <- hunspell(master$answer, dict = dictionary("en_US"))
```

The result from the `hunspell()` function is a list object of spelling errors for each row of data. For example, when responding to *apple*, a participant wrote *fruit grocery store orchard red green yelloe good with peanut butter good with caramell*, and the spelling errors were denoted as *yelloe caramell*. After checking for errors, the `hunspell_suggest()` function was used to determine the most likely replacement for each error.

```
## Check for suggestions
spelling_suggest <- lapply(spelling_errors, hunspell_suggest)
```

For *yelloe*, both *yellow yell* were suggested, and *caramel caramels caramel l camellia camel* were suggested for *caramell*. The suggestions are presented in most probable order, and using a few loops with the substitute (`gsub()`) function, we can replace all errors with the most likely replacement in a new dataset `spell_checked`. A specialized dictionary with pre-coded error responses and corrections could be implemented at this stage. Other paid alternatives, such as Bing Spell Check, can be a useful avenue for datasets that may contain brand names (i.e, *apple* versus *Apple*) or slang terms and provides context sensitive corrections (e.g., keeping *Apple* as a response to computer, but not as a response to green).

```
## Replace with most likely suggestion
spell_checked <- master
### Loop over the dataframe
for (i in 1:nrow(spell_checked)){
  ### See if there are spelling errors
  if (length(spelling_errors[[i]]) > 0) {
    ### Loop over all errors
    for (q in 1:length(spelling_errors[[i]])){
      ### Replace with the first answer
      spell_checked$answer[i] <- gsub(spelling_errors[[i]][q],
                                       spelling_suggest[[i]][[q]][1],
                                       spell_checked$answer[i])
    }
  }
}
```

## Lemmatization

The next step approaches the grouping different word forms that share the same lemma. The process of lemmatizing words involves using a lexeme set (i.e., all words forms that have the same meaning, *am, are, is*) to convert into a common lemma (i.e., *be*) from a trained dictionary. In contrast, stemming involves processing words using heuristics to remove affixes or inflections, such as *ing* or *s*. The stem or root word may not reflect an actual word in the language, as simply removing an affix does not necessarily produce the lemma. For example, in response to *airplane*, *flying* can be easily converted to *fly* by removing the *ing* inflection. However, this same heuristic converts the feature *wings* into *w* after removing both the *s* for a plural marker and the *ing* participle marker.

Lemmatization is the likely choice for processing property norms, and this process can be achieved by installing `TreeTagger` (Schmid, 1994) and the `koRpus` package in *R* (Michalke, 2018). TreeTagger is a trained tagger designed to annotate part of speech and lemma information in text, and parameter files are available for multiple languages. The koRpus package includes functionality to use TreeTagger in *R*. After installing the package and TreeTagger, we will create a unique set of tokenized words to lemmatize to speed computation.

```
lemmas <- spell_checked
## Install the koRpus package
#install.packages("koRpus")
#install.packages("koRpus.lang.en")
## You must load both packages separately
library(koRpus)
library(koRpus.lang.en)
## Install TreeTagger
#https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
## Find all types for faster lookup
all_answers <- tokenize(lemmas$answer, format = "obj", tag = F)
all_answers <- unique(all_answers)
```

The `treetag()` function calls the installation of TreeTagger to provide part of speech tags and lemmas for each token. Importantly, the `path` option should be the directory of the TreeTagger installation.

```
## This example has both suppressWarnings & suppressMessages
## You should first view these to ensure proper processing
temp_tag <- suppressWarnings(
  suppressMessages(
    ## Note: the NULL option is to control for the <unknown> that appears
    ## to occur with the last word in each text
    treetag(c(all_answers, "NULL"),
            ## Control the parameters of treetagger
```

```r
            treetagger="manual", format="obj",
            TT.tknz=FALSE, lang="en",
            TT.options=list(path="~/TreeTagger", preset="en"))))
```

This function returns a tagged corpus object, which can be converted into a dataframe of the token-lemma information. The goal would be to replace inflected words with their lemmas, and therefore, unknown values, number tags, and equivalent values are ignored by subsetting out these from the dataset. Table 2 portrays the results from TreeTagger.

```r
## Remove all tags not using
replacement_lemmas <- temp_tag@TT.res
replacement_lemmas <- subset(replacement_lemmas,
                             #ignore punctuation
                             wclass != "punctuation" &
                             #unknown values
                             lemma != "<unknown>" &
                             #numbers
                             lemma!= "@card@" &
                             #token should change more than case
                             tolower(token) != tolower(lemma))
```

Similar to spelling correction `stri_replace_all_regex()` is used to replace the wordforms with their corresponding lemmas from the `stringi` package (Gagolewski & Tartanus, 2019). Table 3 shows the processed data at this stage.

```r
## Install the stringi package
#install.packages("stringi")
library(stringi)
## Replace all the original tokens with new lemmas using \\b for word boundaries
lemmas$answer <- stri_replace_all_regex(str = lemmas$answer,
                        pattern = paste("\\b", replacement_lemmas$token, "\\b", sep = ""),
                        replacement = replacement_lemmas$lemma,
                        vectorize_all = F, list(case_insensitive = TRUE))
```

## Multi-word Sequences

Multi-word sequences are often coded to mimic a Collins and Quillian (1969) style model, with "is-a" and "has-a" type markers. If data were collected to include these markers, this step would be pre-encoded into the output data, rendering the following code unnecessary. A potential solution for processing messy data could be to search for specific part of speech sequences that mimic the "is-a" and "has-a" strings, and a more complex set of regular expressions has been implented in Strudel by Baroni, Murphy, Barbu, and Poesio (2010). An examination of the coding in McRae et al. (2005) and Devereux et al. (2014) indicates that the feature tags are often verb-noun or verb-adjective-noun sequences. Using TreeTagger on each concept's answer set, we can obtain the parts of speech in context for each lemma. With `dplyr` (Wickham, Francios, Henry, Muller, & Rstudio, 2019), new columns are added to tagged data to show all bigram and trigram sequences. All verb-noun and verb-adjective-noun combinations are selected, and any words not part of these multi-word sequences are treated as unigrams. Finally, the `table()` function is used to tabulate the final count of n-grams and their frequency.

```
## Create an empty dataframe
multi_words <- data.frame(Word=character(),
                          Feature=character(),
                          Frequency=numeric(),
                          stringsAsFactors=FALSE)
## Create unique word list to loop over
unique_concepts <- unique(lemmas$word)
## Install dplyr
#install.packages("dplyr")
library(dplyr)
## Loop over each word
for (i in 1:length(unique_concepts)){
  ## Create parts of speech for clustering together
  temp_tag <- suppressWarnings(
    suppressMessages(
      treetag(c(lemmas$answer[lemmas$word  == unique_concepts[i]], "NULL"),
          ## Control the parameters of treetagger
```

```
        treetagger="manual", format="obj",
        TT.tknz=FALSE, lang="en",
        TT.options=list(path="~/TreeTagger", preset="en"))))
## Save only the dataframe, remove NULL
temp_tag <- temp_tag@TT.res[-nrow(temp_tag@TT.res) , ]
## Subset out information you don't need
temp_tag <- subset(temp_tag,
                   wclass != "comma" & wclass != "determiner" &
                     wclass != "preposition" & wclass != "modal" &
                     wclass != "predeterminer" & wclass != "particle" &
                     wclass != "to" & wclass != "punctuation" &
                     wclass != "fullstop" & wclass != "conjunction" &
                     wclass != "pronoun")
## Create a temporary tibble
temp_tag_tibble <- as_tibble(temp_tag)
## Create part of speech and features combined
temp_tag_tibble <- mutate(temp_tag_tibble,
                          two_words = paste(token,
                                            lead(token), sep = "_"))
temp_tag_tibble <- mutate(temp_tag_tibble,
                          three_words = paste(token,
                                              lead(token), lead(token, n = 2L),
                                              sep = "_"))
temp_tag_tibble <- mutate(temp_tag_tibble,
                          two_words_pos = paste(wclass,
                                                lead(wclass), sep = "_"))
temp_tag_tibble <- mutate(temp_tag_tibble,
                          three_words_pos = paste(wclass,
                                                  lead(wclass), lead(wclass, n = 2L),
                                                  sep = "_"))
## Find adjective, noun, verb combinations to cluster on
verb_nouns <- grep("\\bverb_noun", temp_tag_tibble$two_words_pos)
adj_nouns <- grep("\\badjective_noun", temp_tag_tibble$two_words_pos)
verb_adj_nouns <- grep("\\bverb_adjective_noun", temp_tag_tibble$three_words_pos)
## Use combined and left over features
features_for_table <- c(temp_tag_tibble$two_words[verb_nouns],
                        temp_tag_tibble$two_words[adj_nouns],
                        temp_tag_tibble$three_words[verb_adj_nouns],
                        temp_tag_tibble$token[-c(verb_nouns, verb_nouns+1,
                                                 adj_nouns, adj_nouns+1,
                                                 verb_adj_nouns, verb_adj_nouns+1,
```

```
                                                verb_adj_nouns+2)])
  ## Create a table of frequencies
  word_table <- as.data.frame(table(features_for_table))
  ## Clean up the table
  word_table$Word <- unique_concepts[i]
  colnames(word_table) = c("Feature", "Frequency", "Word")
  multi_words <- rbind(multi_words, word_table[ , c(3, 1, 2)])
}
```

This procedure produces mostly positive output, such as *fingers-have_fingernails* and *couches-have_cushions*. One obvious limitation is the potential necessity to match this coding system to previous codes, which were predominately hand processed. Further, many similar phrases, such as the ones for *zebra* shown below may require fuzzy logic matching to ensure that the different codings for *is-a-horse* are all combined together, as shown in Table 4.

## Bag of Words

The bag of words approach simply treats each token as a separate feature to be tabulated for analysis. After stemming and lemmatization, the data can be processed as single word tokens into a table of frequencies for each cue word. The resulting dataframe is each cue-feature combination with a total for each feature.

```
## Create an empty dataframe
bag_words <- data.frame(Word=character(),
                        Feature=character(),
                        Frequency=numeric(),
                        stringsAsFactors=FALSE)
## Loop over each word
for (i in 1:length(unique_concepts)){
  ## Create a table of frequencies
  word_table <- as.data.frame(table(
    ## Tokenize the words
    tokenize(
      ## Put all answers together in one character string
```

```
      paste0(lemmas$answer[lemmas$word == unique_concepts[i]], collapse = " "),
      format = "obj", tag = F)))

  ## Clean up the table
  word_table$Word <- unique_concepts[i]
  colnames(word_table) = c("Feature", "Frequency", "Word")

  bag_words <- rbind(bag_words, word_table[ , c(3, 1, 2)])
}
## Remove punctuation
bag_words <- bag_words[-c(grep('^[[:punct:]]',bag_words$Feature)), ]
```

Table 5 shows the top ten most frequent responses to *zebra* given the bag of words approach. The top ten features in zebra indicate a match to the multi-word sequence approach but the inclusion of words such as *be, in, a* indicate the need to remove irrelevant words listed with features.

## Stopwords

As shown in Figure 1, the next stage of processing would be to exclude stopwords, such as *the, of, but*, for either the multi-word sequence or bag of word style processing. The `stopwords` package (Benoit, Muhr, & Watanabe, 2017) includes a list of stopwords for more than 50 languages. For multi-word sequence processing, these values can be removed by subsetting the data to exclude stopwords as unigrams.

```
## Install the stopwords package or use tm
#install.packages("stopwords")
library(stopwords)
## Remove stop words from either processing approach
multi_words_nostop <- subset(multi_words,
                      !(Feature %in% stopwords(language = "en",
                                               source = "snowball")))
bag_words_nostop <- subset(bag_words,
                    !(Feature %in% stopwords(language = "en",
                                             source = "snowball")))
```

## Descriptive Statistics

The finalized data now represents a a processed set of cue-feature combinations with their frequencies for analysis. Given the differences in sample size across data collection points from Buchanan et al. (2019), this information was merged with the sample data. Table 6 includes descriptive statistics for the processed cue-feature set. First, the number of cue-feature combinations was calculated by taking the average number of cue-feature listings for each cue. Therefore, the total number of features listed for *zebra* might be 100, while *apple* might be 45, and these values were averaged.

More cue-feature combinations are listed for the multi-word approach, due to differences in combinations for some overlapping features as shown in Table 4. The large standard deviation for both approaches indicates that cues have a wide range of possible features listed. The correlation provided represents the relation between sample size for a cue and the number of features listed for that cue. These values are high and positive, indicating that the number of unique features increases with each participant. Potentially, many of the cue-feature combinations could be considered idiosyncratic. The next row of the table denotes the average number of cue-feature responses listed by less than 10% of the participants. This percent of responses is somewhat arbitrary, as each researcher has determined where the optimal criterion should be. For example, McRae et al. (2005) used 16% or 5/30 participants as a minimum standard, and Buchanan et al. (2019) recently used a similar criteria. A large number of cue-features are generated by a small number of participants, indicating that these are potentially idiosyncratic or part of long tailed distribution of feature responses with many low frequency features. The advantage to the suggested data processing pipeline and code provided here is the ability of each researcher to determine their own level of response necessary, if desired. Additionally, feature weighting using statistics such as pointwise mutual information could be implemented to discount rare features without excluding them.

The next two lines of Table 6 indicate cue-feature combination frequencies, such as the number of times *zebra-stripes* or *apple-red* were listed by participants. The percent of responses is the frequency divided by sample size for each cue, to normalize over different sample sizes present in the data. These average frequency/percent was calculated for each cue, and then averaged over all cues. The correlation represents the average frequency/percent for each cue related to the sample size for that cue. These frequencies are low, matching the results for a large number of idiosyncratic responses. The correlation between frequency of response and sample size is positive, indicating that larger sample sizes produce items with larger frequencies. Additionally, the correlation between percent of response and sample size is negative, suggesting that larger sample sizes are often paired with more items with smaller percent likelihoods. Figure 2 displays the correlations for the average cue-frequency responses and the percent cue-frequency responses by sample size. It appears that the relationship between sample size and percent is likely curvilinear, rather than linear. The size of the points indicates the variability (standard deviation of each cue word's average frequency or percent). Variability appears to increase linearly with sample size for average frequency, however, it is somewhat mixed for average percent.

## Internal Comparison of Approach

In this section, we show that the bag of words approach processed completely through code matches a bag of words approach that was hand coded from Buchanan et al. (2019). In Buchanan et al. (2019), the McRae et al. (2005) and Vinson and Vigliocco (2008) datasets were recoded in a bag of words approach, and the comparison between all three is provided below. The multi-word sequence approach would be comparable if one or more datasets used the same structured data collection approach or with considerable hand coded rules for feature combinations. The data from open ended responses, such as the Buchanan et al. (2019), could potentially be compared in the demonstrated multi-word sequence approach, if the raw data from other such projects were available.

Cosine is often used as a measure of semantic similarity, indicating the feature overlap between two sets of cue-feature lists. These values can range from 0 (no overlap) to 1 (perfect overlap). There are two potential cosine values from the Buchanan et al. (2019): the raw cosine, which included all features as listed without lemmatization or stemming, and the translated cosine, which included hand lemmatization processing. Each cue in the sample data for this project was compared to the corresponding cue in the Buchanan et al. (2019). If data were processed in an identical fashion, the cosine values would be nearly 1 for Buchanan et al. (2019) data or match the cosine values found for McRae et al. (2005) and Vinson and Vigliocco (2008) in the Buchanan et al. (2019) results (original feature cosine = .54-.55, translated features = .66-.67). However, all previous datasets have been reduced by eliminating idiosyncratic features at various points, and therefore, we might expect that noise in this data to reduce the average cosine values. Table 7 indicates the cosine values for each cue paired with itself in different scenarios. On the left, the cosine values with stopwords are provided for both the original feature listed (i.e., no lemmatization) and the translated feature (i.e., hand lemmatization). The right side of the table includes the cosine values once stopwords have been removed. The removal of stopwords increases the match between sets indicating how removing these terms can improve comparison and quality. The cosine values for no stopwords indicate a somewhat comparable set of data, with lower values for McRae et al. (2005) than previous results in the original feature sets. These values indicate that the data processed entirely in *R* produces a comparable set of results, albeit with added noise of small frequency features.

## External Comparison of Approach

The MEN dataset (Bruni et al., 2014) contains cue-cue pairs of English words rating for similarity by Amazon Mechanical Turk participants for stimuli taken from the McRae et al. (2005) feature norms. In their rating task, participants were shown two cue-cue pairs and asked to select the more related pair of the two presented. Each pair was rated by 50 participants, and thus, a score of 50 indicates high relatedness, while a score of 0 indicates no relatedness. The ratings for the selected set of cues provided in this analysis was 2 - 49 with an average rating of 25.79 ($SD$ = 12.00). The ratings were compared to the cosine calculated between cues using the bag of words method with and without stopwords. The correlation between bag of words cosines with stopwords and the MEN ratings was $r$ = .54, 95% CI [.42, .63], $N$ = 179, indicating agreement between raters and cosine values. The agreement between ratings and bag of word cosine values was higher when stopwords were excluded, $r$ = .69, 95% CI [.61, .76].

## Future Directions

An attractive property of the subjective feature listing task is that it results in transparent representations. As a result, many researchers have taken additional steps to group specific types of knowledge together, depending on semantic relations (e.g., taxonomy relations) or their mapping onto distinct brain regions (**???**). Typically this involves applying a hand-crafted coding scheme, which requires a substantial effort. One of the common ontologies is the one developed by Wu and Barsalou (2009). The ontology is structured as a hierarchical taxonomy for coding categories as part of the feature listing task. It has been used in several projects, notably the McRae et al. (2005). Examples of the categories include taxonomic (synonyms, subordinates), entity (internal components, behavior, spatial relations), situation (location, time), and introspective properties (emotion, evaluation). Coding ontology may be best performed systematically with look-up rules of previously decided upon factors, however, clustering analyses may provide a potential avenue to explore categorizing features within the current dataset. One limitation to this method the sheer size of the idiosyncratic features as mentioned above, and thus, features smaller in number may be more difficult to group.

Potentially, simple ontology can be mapped using results from Strudel (structured dimension extraction and labeling, Baroni et al., 2010). Strudel is a corpus-based semantic model wherein cue words are found in a large text corpus and matched to nouns, verbs, and adjectives that appear near a concept. Using specific patterns of expected feature listing, Baroni et al. (2010) were able build a model of English concepts and their properties that aligned with semantic feature production norms. From this model, they were able to cluster properties based on their lexical patterns. For example, if a sentence included the phrase *fruit, such as an apple*, this lexical pattern would be classified as *such_as+right*, indicating that the concept (apple) was found to the right of the property (fruit) with the phrase such as connecting them. Using clustering, Baroni et al. (2010) was able to assign four ontology labels to properties: part, category, location, and function. Using these results, we can match 2259 of the bag of words features (5%). These features were predominately parts (39.9), followed by function (30.5), location (24.0), and category (5.5). Table 8 indicates ten of the most frequent cue-feature pairs for each ontology label, excluding duplicate features across cues. An examination of the top results indicates coherent labels (parts: *zebra-stripe*, location: *shoe-foot*, and category: *furniture-table*); however, there are also a few mismatches (location: *scissors-cut*, function: *leaf-green*). This model represents an area in which one might begin to automate the labeling process, likely combined with other pre-defined rulesets.

## Discussion

- this sort of thing is great for replication purposes, which is pretty important because of the garden of forking paths which applies not just to statistical analyses but also to processing.
- we've provided a workflow suggestion that a researcher can use to format their work,

along with functions that can be detailed to match any hand processing results.

- weave this to match introduction
- how concrete or abstract the words are

## References

Ashcraft, M. H. (1978). Property norms for typical and atypical items from 17 categories: A description and discussion. *Memory & Cognition*, *6*(3), 227–232. doi:10.3758/BF03197450

Baroni, M., Murphy, B., Barbu, E., & Poesio, M. (2010). Strudel: A Corpus-Based Semantic Model Based on Properties and Types. *Cognitive Science*, *34*(2), 222–254. doi:10.1111/j.1551-6709.2009.01068.x

Benoit, K., Muhr, D., & Watanabe, K. (2017). stopwords: Multilingual Stopword Lists. Retrieved from https://cran.r-project.org/web/packages/stopwords/index.html

Bruni, E., Tran, N. K., & Baroni, M. (2014). Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, *49*, 1–47. doi:10.1613/jair.4135

Buchanan, E. M., Holmes, J. L., Teasley, M. L., & Hutchison, K. A. (2013). English semantic word-pair norms and a searchable Web portal for experimental stimulus creation. *Behavior Research Methods*, *45*(3), 746–757. doi:10.3758/s13428-012-0284-z

Buchanan, E. M., Valentine, K. D., & Maxwell, N. P. (2019). English semantic feature production norms: An extended database of 4436 concepts. *Behavior Research Methods*. doi:10.3758/s13428-019-01243-z

Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, *8*(2), 240–247. doi:10.1016/S0022-5371(69)80069-1

Devereux, B. J., Tyler, L. K., Geertzen, J., & Randall, B. (2014). The Centre for Speech, Language and the Brain (CSLB) concept property norms. *Behavior Research Methods*, *46*(4), 1119–1127. doi:10.3758/s13428-013-0420-4

Gagolewski, M., & Tartanus, B. (2019). stringi: Character String Processing Facilities. Retrieved from https://cran.r-project.org/web/packages/stringi/index.html

Kremer, G., & Baroni, M. (2011). A set of semantic norms for German and Italian. *Behavior Research Methods*, *43*(1), 97–109. doi:10.3758/s13428-010-0028-x

Lenci, A., Baroni, M., Cazzolli, G., & Marotta, G. (2013). BLIND: A set of semantic feature norms from the congenitally blind. *Behavior Research Methods*, *45*(4), 1218–1233. doi:10.3758/s13428-013-0323-4

McRae, K., Cree, G. S., Seidenberg, M. S., & McNorgan, C. (2005). Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, *37*(4), 547–559. doi:10.3758/BF03192726

Michalke, M. (2018). koRpus: An R Package for Text Analysis. Retrieved from https://cran.r-project.org/web/packages/koRpus/index.html

Montefinese, M., Ambrosini, E., Fairfield, B., & Mammarella, N. (2013). Semantic memory: A feature-based analysis and new norms for Italian. *Behavior Research Methods*, *45*(2), 440–461. doi:10.3758/s13428-012-0263-4

Ooms, J. (2018). The hunspell package: High-Performance Stemmer, Tokenizer, and Spell Checker for R. Retrieved from https://cran.r-project.org/web/packages/hunspell/vignettes/intro.html{\#}setting{_}a{_}language

Reverberi, C., Capitani, E., & Laiacona, E. (2004). Variabili semantico lessicali relative a tutti gli elementi di una categoria semantica: Indagine su soggetti normali italiani per la categoria “frutta". *Giornale Italiano Di Psicologia*, *31*, 497–522.

Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, *7*(4), 573–605. doi:10.1016/0010-0285(75)90024-9

Ruts, W., De Deyne, S., Ameel, E., Vanpaemel, W., Verbeemen, T., & Storms, G. (2004). Dutch norm data for 13 semantic categories and 338 exemplars. *Behavior Research Methods, Instruments, & Computers*, *36*(3), 506–515. doi:10.3758/BF03195597

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. doi:10.1.1.28.1139

Stein, L., & de Azevedo Gomes, C. (2009). Normas Brasileiras para listas de palavras associadas: Associação semântica, concretude, frequência e emocionalidade. *Psicologia: Teoria E Pesquisa*, *25*, 537–546. doi:10.1590/S0102-37722009000400009

Toglia, M. P. (2009). Withstanding the test of time: The 1978 semantic word norms. *Behavior Research Methods*, *41*(2), 531–533. doi:10.3758/BRM.41.2.531

Toglia, M. P., & Battig, W. F. (1978). *Handbook of semantic word norms.* Hillside, NJ: Earlbaum.

Vinson, D. P., & Vigliocco, G. (2008). Semantic feature production norms for a large set of objects and events. *Behavior Research Methods*, *40*(1), 183–190. doi:10.3758/BRM.40.1.183

Vivas, J., Vivas, L., Comesaña, A., Coni, A. G., & Vorano, A. (2017). Spanish semantic feature production norms for 400 concrete concepts. *Behavior Research Methods*, *49*(3), 1095–1106. doi:10.3758/s13428-016-0777-2

Wickham, H., Francios, R., Henry, L., Muller, K., & Rstudio. (2019). dplyr: A Grammar of Data Manipulation. Retrieved from https://cloud.r-project.org/web/packages/dplyr/index.html

Wu, L.-l., & Barsalou, L. W. (2009). Perceptual simulation in conceptual combination: Evidence from property generation. *Acta Psychologica*, *132*(2), 173–189.

doi:10.1016/j.actpsy.2009.02.002

Table 1

*Example of Data Formatted for Tidy Data*

| word | answer |
|---|---|
| airplane | you fly in it its big it is fast they are expensive they are at an airport you have to be trained to fly it there are lots of seats they get very high up |
| airplane | wings engine pilot cockpit tail |
| airplane | wings it flys modern technology has passengers requires a pilot can be dangerous runs on gas used for travel |
| airplane | wings flys pilot cockpit uses gas faster travel |
| airplane | wings engines passengers pilot(s) vary in size and color |
| airplane | wings body flies travel |

Table 2

*Lemma and Part of Speech Information from TreeTagger*

| token | tag | lemma | lttr | wclass |
|---|---|---|---|---|
| is | VBZ | be | 2 | verb |
| are | VBP | be | 3 | verb |
| trained | VBN | train | 7 | verb |
| lots | NNS | lot | 4 | noun |
| seats | NNS | seat | 5 | noun |
| wings | NNS | wing | 5 | noun |

Table 3

*Original Data with Lemmatization*

| word | answer |
|---|---|
| airplane | you fly in it its big it be fast they be expensive they be at an airport you have to be train to fly it there be lot of seat they get very high up |
| airplane | wing engine pilot cockpit tail |
| airplane | wing it fly modern technology have passenger require a pilot can be dangerous run on gas use for travel |
| airplane | wing fly pilot cockpit use gas fast travel |
| airplane | wing engine passenger pilot(s) vary in size and color |
| airplane | wing body fly travel |

Table 4

*Multi-Word Sequence Examples for Zebra*

| Word | Feature | Frequency |
|---|---|---|
| zebra | be_horse | 1 |
| zebra | be_similar_horse | 1 |
| zebra | build_horse | 1 |
| zebra | fast_horse | 1 |
| zebra | horse | 19 |
| zebra | horse-like | 1 |
| zebra | look_similar_horse | 1 |
| zebra | related_horse | 1 |
| zebra | resemble_small_horse | 1 |
| zebra | run_fast_horse | 1 |

Table 5

*Bag of Words Examples for Zebra*

| Word | Feature | Frequency |
|---|---|---|
| zebra | stripe | 71 |
| zebra | black | 63 |
| zebra | white | 61 |
| zebra | be | 56 |
| zebra | animal | 54 |
| zebra | have | 54 |
| zebra | a | 46 |
| zebra | and | 46 |
| zebra | in | 41 |
| zebra | horse | 32 |

Table 6

*Descriptive Statistics of Text Processing Style*

| | Multi-Word Sequences | | | Bag of Words | | |
|---|---|---|---|---|---|---|
| Statistics | *M* | *SD* | *r* | *M* | *SD* | *r* |
| Number of Cue-Features | 212.92 | 115.63 | 0.77 | 171.80 | 76.96 | 0.66 |
| Frequency of Idiosyncratic Response | 205.86 | 114.20 | 0.78 | 158.85 | 73.97 | 0.69 |
| Frequency of Cue-Feature Response | 1.80 | 2.61 | 0.75 | 2.73 | 4.80 | 0.83 |
| Percent of Cue-Feature Response | 2.95 | 3.88 | -0.66 | 4.34 | 4.80 | -0.62 |

*Note*. Correlation represents the relation between the statistic listed for that row and the sample size for the cue.

Table 7

*Cosine Overlap with Previous Data Collection*

| | With Stopwords | | No Stopwords | |
|---|---|---|---|---|
| Statistic | Original | Translated | Original | Translated |
| B Mean | .54 | .57 | .69 | .72 |
| B SD | .16 | .17 | .17 | .16 |
| M Mean | .32 | .48 | .38 | .58 |
| M SD | .15 | .14 | .18 | .14 |
| V Mean | .50 | .49 | .59 | .58 |
| V SD | .18 | .19 | .18 | .19 |

*Note.* Translated values are hand coded lemmatization from Buchanan et al. (2019). B: Buchanan et al. (2019), M: McRae et al. (2005), V: Vinson & Vigliocco (2008). *N* values are 226, 61, and 68 respectively.

Table 8

*Top Ten Ontology Labels*

| Parts | Function | Location | Category |
|---|---|---|---|
| brush use | brush hair | scissors cut | flute instrument |
| lawn grass | river water | snow cold | snow white |
| snail shell | branch tree | farm land | elephant animal |
| river stream | chair sit | cabin wood | cabbage green |
| radio music | leaf plant | rocket space | dagger knife |
| elephant trunk | kitchen food | breakfast day | apple fruit |
| door open | hammer nail | stone rock | hammer tool |
| zebra stripe | oven cook | bacon pig | lion king |
| river flow | garden flower | shoe foot | cabbage vegetable |
| dragon fire | leaf green | tree leaf | furniture table |

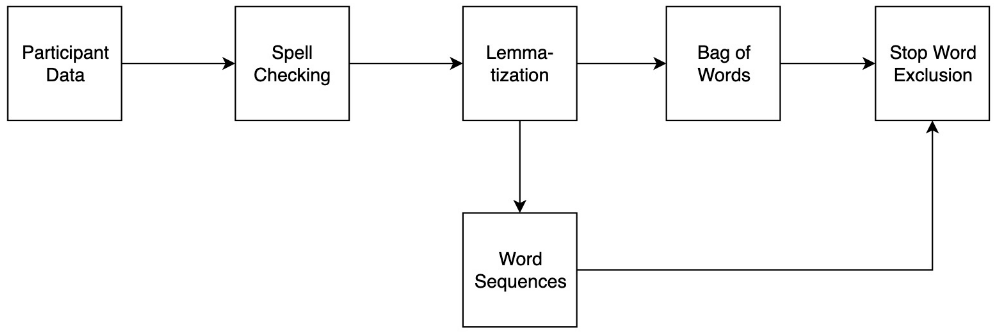


*Figure 1*. Flow chart illustrating how feature listings are recoded to obtain a standard feature format.

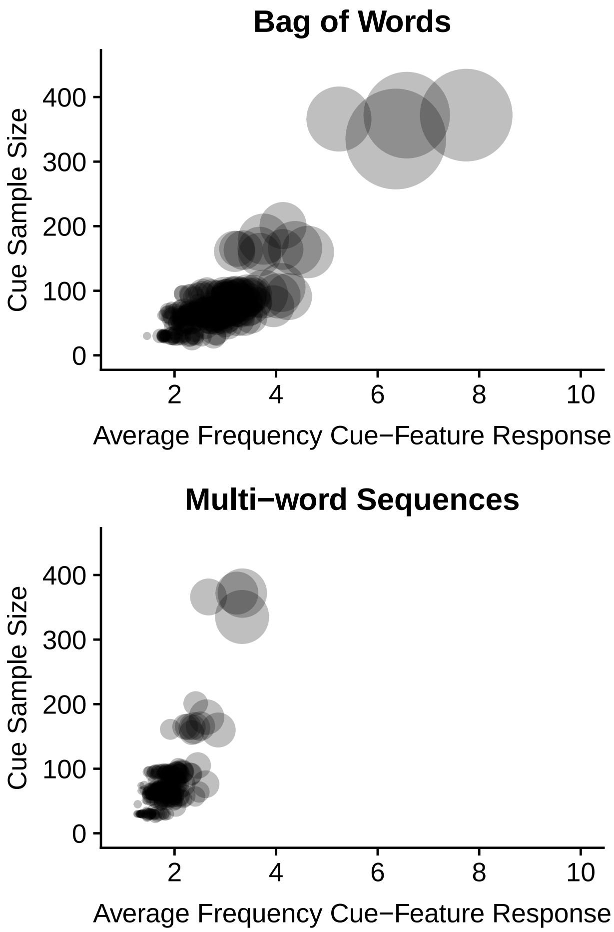


*Figure 2*. Correlation of sample size with the average cue-feature frequency (left) and percent (right) of response for each cue for both processing approaches. Each point represents a cue word, and the size of the point indicates the variability of the average frequency (left) or percent (right).