A practical primer on processing semantic property norm data

|  |
| --- |
| Erin M. Buchanan1, Simon De Deyne2, & Maria Montefinese3 |
| 1 Harrisburg University of Science and Technology |
| 2 University of Melbourne |
| 3 University of Padua |
|  |

# Author note

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

Word count:

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.

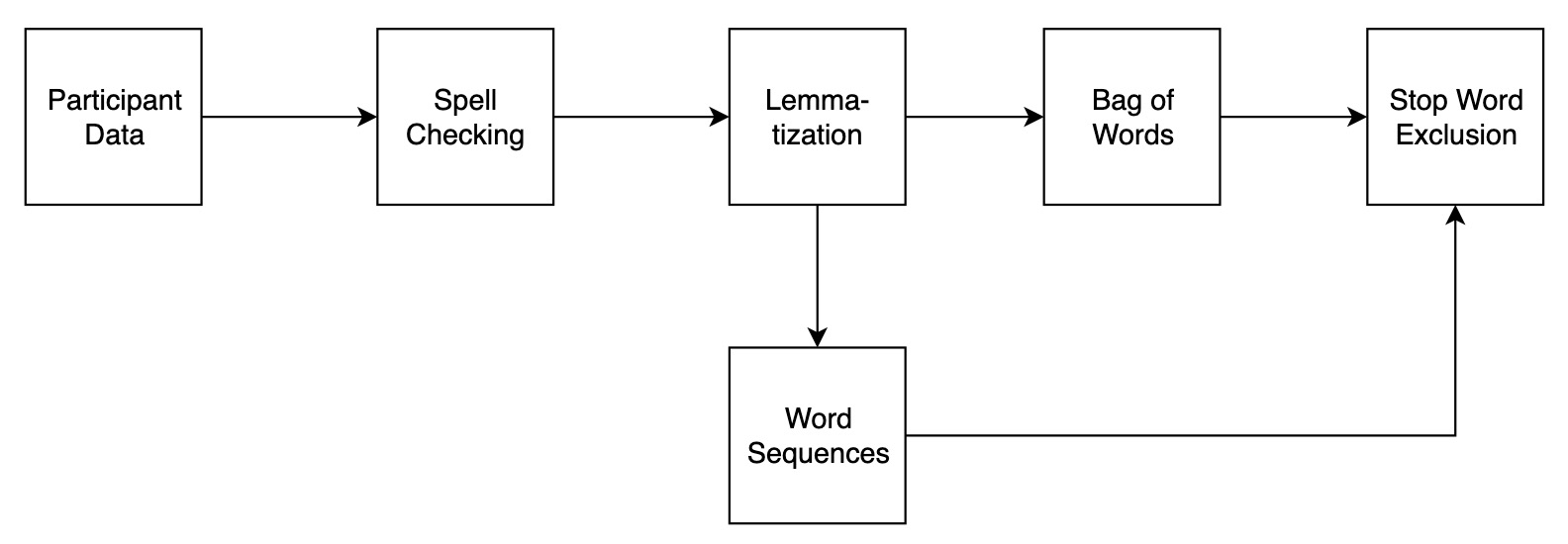
Define concept, feature for clarity throughout - make sure you use these two terms consistently.

1. Pointers about how to collect the data
2. 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.

1. Typical operations performed on features



(#fig:flow\_chart)Flow chart of proposed semantic processing feature steps.

In the next several sections, we provide a tutorial using *R* on how data from the semantic property norm task might be processed from raw input to finalized output. Figure @ref(fig:flow\_chart) 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 17177 unique concept-feature responses for 226 concepts from Buchanan et al. (2019) that were included in McRae et al. (2005), Vinson and Vigliocco (2008), and Bruni, Tran, and Baroni (2014). The data should be structured in tidy format wherein each concept-feature observation is a row and each column is a variable (Wickham, 2014). Therefore, the data includes a word column with the normed concept and an answer column with the participant answer.

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), which is the spell checking library used in popular programs such as FireFox, Chrome, RStudio, and OpenOffice. 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.

## 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 precoded 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.

## 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 clustering of word forms into their lemma or head word from a dictionary. 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 langauge, 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. Several packages for *R* include customizable stemmers, notably the hunspell, corpus (Perry, 2017), and tm (Feinerer, Hornik, & Artifex Software, 2018) packages.

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 langauges. 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 function 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  
 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 subseting out these from the dataset.

## 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))

From this dataset, you can use the stringi package (Gagolewski & Tartanus, 2019) to replace all of the original tokens with their lemmas. This package allows for replacement lookup across a large set of subsitutions.

## 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))

## 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. 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$word == unique\_concepts[i]], "NULL"),   
 ## Control the parameters of treetagger  
 treetagger="manual", format="obj",  
 TT.tknz=FALSE, lang="en",  
 TT.options=list(path="~/TreeTagger", preset="en"))))  
 ## Save only the dataframe, remove NULL  
 temp\_tag <- temp\_tag@TT.res[-nrow(temp\_tag@TT.res) , ]  
 ## Subset out information you don't need  
 temp\_tag <- subset(temp\_tag,   
 wclass != "comma" & wclass != "determiner" &   
 wclass != "preposition" & wclass != "modal" &  
 wclass != "predeterminer" & wclass != "particle" &  
 wclass != "to" & wclass != "punctuation" &   
 wclass != "fullstop" & wclass != "conjunction" &   
 wclass != "pronoun")  
 ## Create a temporary tibble   
 temp\_tag\_tibble <- as\_tibble(temp\_tag)  
 ## Create part of speech and features combined  
 temp\_tag\_tibble <- mutate(temp\_tag\_tibble,   
 two\_words = paste(token,   
 lead(token), sep = "\_"))  
 temp\_tag\_tibble <- mutate(temp\_tag\_tibble,   
 three\_words = paste(token,   
 lead(token), lead(token, n = 2L),   
 sep = "\_"))  
 temp\_tag\_tibble <- mutate(temp\_tag\_tibble,   
 two\_words\_pos = paste(wclass,   
 lead(wclass), sep = "\_"))  
 temp\_tag\_tibble <- mutate(temp\_tag\_tibble,   
 three\_words\_pos = paste(wclass,   
 lead(wclass), lead(wclass, n = 2L),   
 sep = "\_"))  
 ## Find verb noun or verb adjective nouns to cluster on   
 verb\_nouns <- grep("\\bverb\_noun", temp\_tag\_tibble$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$three\_words[verb\_adj\_nouns],  
 temp\_tag\_tibble$token[-c(verb\_nouns, verb\_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.

## 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.

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 @ref(fig:flow\_chart), 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 (**???**) includes a list of stopwords for more than 50 languages. For multi-word sequence processing, these values can be removed by subseting 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 <- subset(multi\_words,   
 !(Feature %in% stopwords(language = "en",   
 source = "snowball")))  
  
bag\_words <- subset(bag\_words,   
 !(Feature %in% stopwords(language = "en",   
 source = "snowball")))

## Descriptive Statistics

## [1] 0.9836455

## [1] 0.7351418

## [1] 0.6603825

## [1] 0.9163408

## [1] 0.7155022

## [1] 0.8167169

make a table here of the stuff talk about deleting low features or not d. identify cut off for idiosyncratic features (should it be necessary?)

## Internal Comparison of Approach

Compare this data processing to hand processed data from B2019

## raw\_b raw\_m raw\_v translated\_b translated\_m   
## 0.6899807 0.3764903 0.5926762 0.7243672 0.5753183   
## translated\_v   
## 0.5817829

## External Comparison of Approach

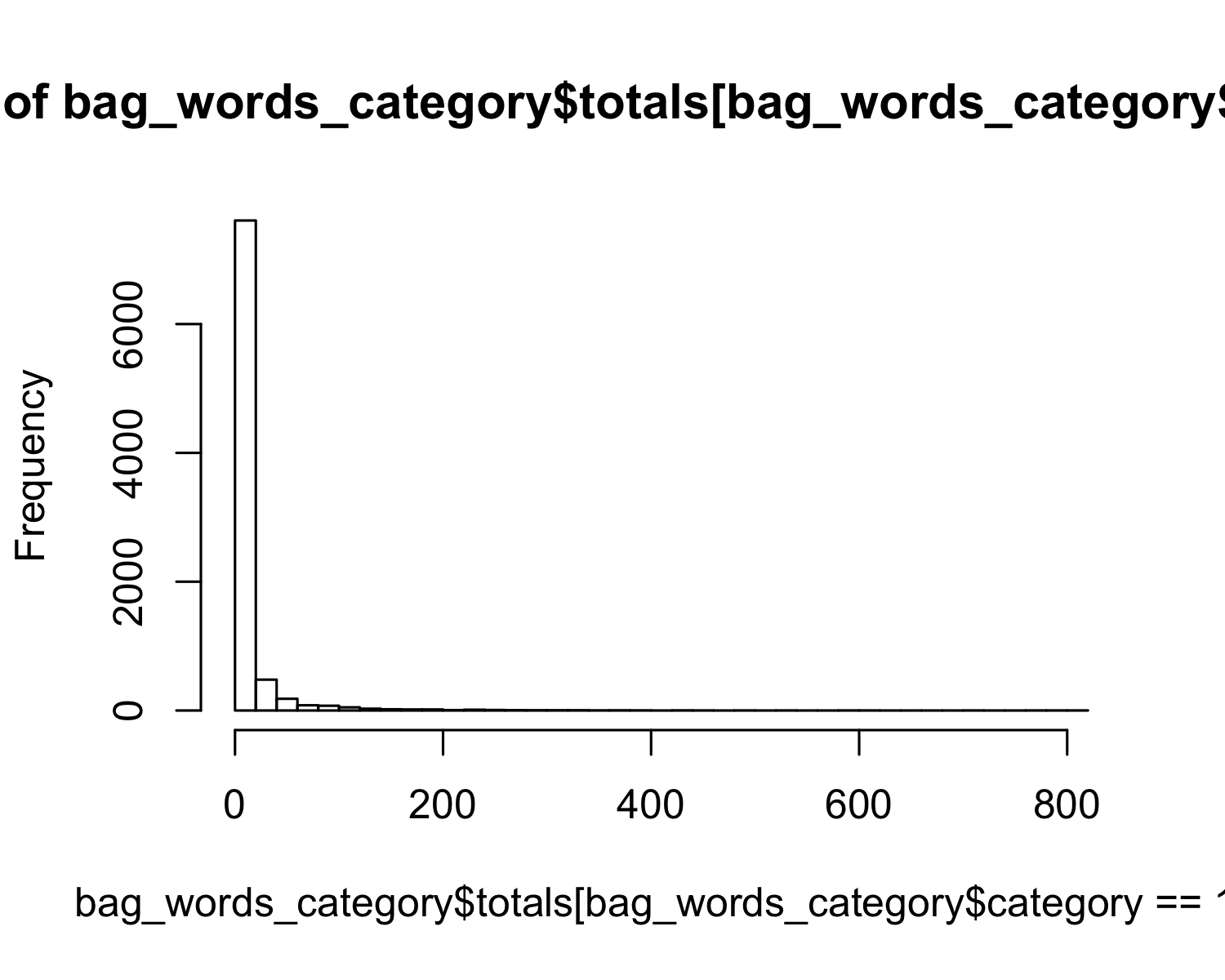
Compare to the MEN dataset

## [1] 0.6889117

## Ontology and Categorization

## [1] 0.9477885 0.8802228 0.8768643 0.8775418 0.8776403 0.8761919 0.8762623  
## [8] 0.8723146 0.8723403 0.8657468 0.8660341 0.8660926 0.8661895 0.8662974  
## [15] 0.8662678 0.8657396 0.8658517 0.8641960 0.8634386 0.8573610 0.8250660  
## [22] 0.8249962 0.8252634 0.8255298 0.7192209 0.7193813 0.7193717 0.7197155  
## [29] 0.7193555 0.7189437 0.7190156 0.7191427 0.7185243

##   
## 1 2 3 4 5 6   
## 8613 17 3 13 11 5



## 1 2 3 4 5 6   
## 11.6864 193.2353 1843.0000 150.6923 124.4545 442.0000

## 1 2 3 4 5 6   
## 36.57800 161.36238 520.74274 97.93313 95.65497 116.06679

## 1 2 3 4 5 6   
## 1 43 1260 70 44 247

## 1 2 3 4 5 6   
## 810 542 2262 383 362 550

# Discussion

# 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](https://doi.org/10.3758/BF03197450)

Bruni, E., Tran, N. K., & Baroni, M. (2014). Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, *49*, 1–47. doi:[10.1613/jair.4135](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.3758/s13428-013-0420-4)

Feinerer, I., Hornik, K., & Artifex Software, I. (2018). tm: Text Mining Package. Retrieved from <https://cran.r-project.org/web/packages/tm/index.html>

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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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>

Perry, P. O. (2017). corpus: Text Corpus Analysis. Retrieved from <http://corpustext.com/>

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](https://doi.org/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](https://doi.org/10.3758/BF03195597)

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. doi:[10.1.1.28.1139](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.3758/s13428-016-0777-2)

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, *59*(10), 1–23. doi:[10.18637/jss.v059.i10](https://doi.org/10.18637/jss.v059.i10)

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>