W(ha/il)t's the (wo/ba)rd?[*]

A cross-sectional analysis of English word pairs that satisfy the properties of the Split Decisions word puzzle

Oliver Daniel

April 30, 2022

Abstract

From three English wordlist corpora, pairs of words are generated which might appear in the Split Decisions word puzzle as published by the New York Times, i.e., at least 5 characters in length and differing by exactly two consecutive characters. The corpora are contrasted among themselves, and then the lexical and phonological properties of those words which form pairs – and those which do not – are analyzed for causative insights. Such insights will be alluded to here once discovered.

Keywords: crosswords, split decisions, computational linguistics

Code available on GitHub (link).

Contents

[*]"What's the word?" or "Wilt's the bard?" (nonsense).

Introduction

About Split Decisions

A few times a year, the New York Times games department releases a new issue of a word puzzle called Split Decisions. Similar to crosswords, the game board consists of boxes arranged in sequence, one for each letter of a word running either left-to-right or top-to-bottom, intersecting with other words at right angles, as shown below in Figure 1. The Split Decisions website (Piscop 2021) attributes the original puzzle format to the late George Bredehorn; after his death, the mantle was taken up by Fred Piscop, whose puzzles appear in the *Times* to this day.

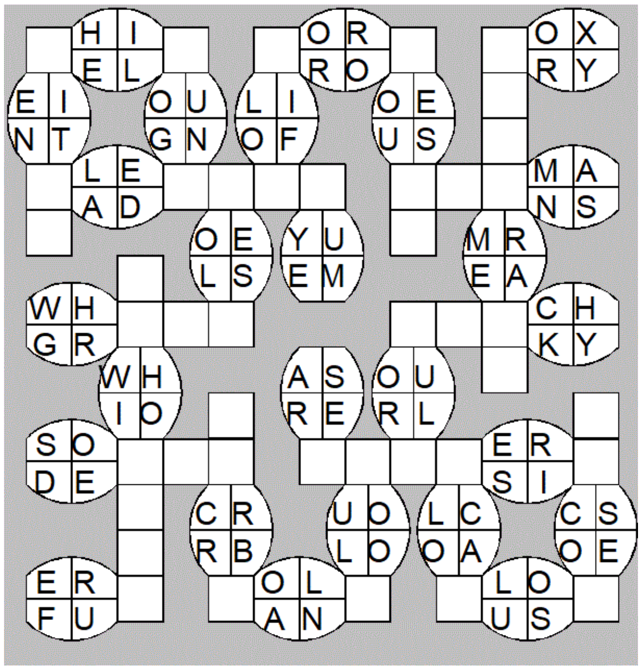


Figure 1: An example Split Decisions puzzle from Piscop's website.

Evidently, instead of cryptic descriptions of correct answers, Split Decisions solvers are only given pairs of two-letter substrings for each word. The remaining boxes, each containing one letter as in a typical crossword, combine with either of these substrings to form an English word. For example, the top-rightmost word, spelled horizontally ("across"), could take a P in its singular empty box to form the words POX and PRY, or an F to form FOX and FRY, potentially among other combinations. However, in filling out the rest of the puzzle, it becomes evident that only the latter solution accommodates both the across word pair, and the down pair that intersects it.

Rules, Notation and Terminology

In order to analyze different aspects of the game, we must first lay out a few rules of how Split Decisions puzzles work in general, as well as some additional constraints we introduce for the purpose of study. Additionally, we lay out some notation and terminology we will use to succinctly describe different segments of a puzzle word.

For lack of a robust LaTeXmechanism for creating authentic-looking Split Decisions clues, like in Figure 1, we will use simpler notation to represent word pairs. Firstly, we will largely use underscores to represent those boxes left unfilled by the puzzle setter, and some visual delineator (often vertical alignment or slashes) to represent the provided two-letter substrings. For certain purposes, we may use a full equation-level LaTeXenvironment, like:

$$\binom{PL}{GR}----$$

But, the above pair may also more succintly be represented in-text by a form like (PL/GR)______ when more appropriate.

A correct fill for the remaining boxes might be (PL/GR)EASE, creating both PLEASE and GREASE, or as ___EASE when the two missing letters are inessential. For the remainder of this paper, such a pair of words will be referred to as a *word pair*, sharing a *common substring*[1] and differing by a *split pair*. The common substring and split pair meet at *boundaries*: using the convention of left-to-right for English, we might describe the above diagram as possessing a *right boundary*, but lacking a left one (as the split begins the word). In the (PL/GR)EASE notation used above, this right boundary is represented by the concatenation of the right parenthesis with another letter. In general, the shorthands *prefix* and *suffix* will be used to refer to common fill before the left boundary and after the right, respectively. Returning to the above example once more, (PL/GR)EASE lacks a prefix and has a suffix of "EASE".

Note that each clued split pair might accept several different fills for the common substring, e.g., (PL/GR)UNGE (PLUNGE, GRUNGE); however, as with crosswords, only one solution will also satisfy all the crossing constraints produced by the rest of the puzzle.

Although a letter may appear on both sides of a split, say, (IT/TA), the letter will *never* appear at the same index in both words, e.g., (IT/AT), as such a pair of words would only differ by a single letter instead of two. Also, although such a puzzle would be fascinating to both solve and analyze, crossing words do *not* go through split pairs, only common letters.

Finally, although a word pair can theoretically be as short as three letters, this paper examines only pairs of length five and above.

Upon understanding the rules of the puzzle, one may start to wonder: what sorts of words form split pairs? Are some splits more common than others? How do the properties of English, as both a spoken and written language, affect the guessability of a given split pair? In this paper, we perform preliminary statistical analysis to investigate these questions, using a variety of computational methods.

Data

Corpora: What's a "word"?

One of the most common side effects of continued study in linguistics is a diminished ability to describe what defines a "word". For the purpose of this study, we have taken three lists

[1]Or *fill*: this part of the puzzle is filled in by the player rather than the constructor.

that purport to contain a wide variety of English words, though certainly not their totality, and used them to generate all possible combinations that produce a valid Split Decisions word pair (in terms of a two-character split pair, etc.). As a result, many entries that appear in one or more of these corpora may seem esoteric, or even offensive, to be cosnidered as words; however, the only predicate for their inclusion in these lists is attested usage, not whether they prescriptively *ought* to be a word. Words appear in these corpora in lexicographical order (i.e., sorted alphabetically), one on each line, in lower-case letters.

The first and smallest corpus comes from an open-source distribution of GNU/Linux, specifically Ubuntu. For brevity, I will usually refer to this corpus as `linux` (note the lower-case letters and monospace font). This corpus, developed over time by online contributors through the development of Ubuntu, is accessed on Linux machines through the symbolic link `/usr/share/dict/words`. It is primarily used as a baseline for other text processing programs' spell-checking and text prediction functionalities.

The `collins` corpus (*Collins Official Scrabble® Words* 2019) is privately produced by HarperCollins LLC, a publishing company well known for producing English- and multiple-language dictionaries. This dictionary in particular finds notable use in competitive Scrabble play: competitors in so-called 'Collins divisions' at tournaments will study from and refer to this list as the superset of all playable Scrabble words.

Similarly, the `nwl` corpus (NASPA Dictionary Committee 2020) was produced directly by members of the North American Scrabble Association (NASPA) for the purpose of tournament play. In Canada and the United States, this word list serves as the *de facto* standard for competitive Scrabble, and is usually the default wordlist unless another is specified, as in the aforementioned Collins divisions.

From each of these corpora, only those entries consisting solely of five or more Latin characters without diacritics were accepted for further processing.

Table 1: Post-processing statistics for each corpus of English words.

	Words		Word pairs	Most frequent (occurrences)		
Corpus	Total	Valid	Total	First split	Second split	Split pair
linux	102774	92243	124332	ck (2603)	st (3135)	ng/on (640)
collins	279496	272384	763072	er (15851)	st (14809)	ng/on (1632)
nwl	191852	186471	434847	er (8996)	st (9008)	ng/on (1235)

Processing & Caching

The algorithm for determining whether two strings of the same length form a Split Decisions pair is quite simple.

1. Traverse the indices of both strings, comparing their characters pairwise. That is, compare the first letters of both strings, then the second, and so on.
2. Record each index at which the two characters differ.
3. Once traversal is complete, if there are exactly two differences and they differ in index by exactly one, then the two strings form a word pair. Otherwise, continue with another combination of strings.

However, initial experiments with identifying these word pairs directly in R (R Core Team 2021) proved to be untenably slow. Since every word in a corpus has to be compared to every other word of the same length at least once, any algorithm[2] used to do so cannot run in better than quadratic time, and even R packages with bindings to C took several hours to run on even the smallest of the corpora. Instead, a Python script was written to process the words more efficiently, and then write the results in a CSV file that could be more readily imported into the R environment. This precomputation phase significantly reduced start-up time when creating R sessions to author this paper, or to knit this document into a PDF using `knitr` (Xie 2021). A link to this script can be found in the Appendix.

Apart from these, the remainder of data processing from this paper was performed in R using the popular `tidyverse` package collection (Wickham et al. 2019), and visualized using the indispensible `ggplot2` (Wickham 2016).

Variables

These cached CSVs accounted for the following variables of each word pair:

1. `length`: the length of each word in the pair. Ranges from 5 to 21.
2. `x`: the lexicographically-first full word in the pair, with no additional notation for the split or any boundaries.
3. `y`: the lexicographically-second full word in the pair.
4. `start`: the 1-index of the beginning of the split. For example, a split at the beginning of a word would have value 1.
5. `end`: The 1-index of the end of the split. This is a convenience variable: each split has a length of exactly two, and hence the ending index is always equal to `start + 1`.
6. `split_x`: The lexicographically-first of the two splits. As a property, the `start`th to `end`th substring of `x` is always this value.
7. `split_y`: The lexicographically-second of the two splits, as above.
8. `corpus`: The corpus from which this pair was determined. One of `linux`, `collins`, or `nwl`.

In certain context-specific scenarios, other variables are computed:

9. `type`: divides each word pair into one of three categories, depending on the positioning of its split pair. If the split pair is at the beginning of the word, the type is "prefix"; if at the end, "suffix"; and otherwise, "middle". (PL/GR)EASE is a "prefix"-type pair, for example.
10. `rank`: when comparing different pairs, splits, etc. by their frequency, a ranking variable is introduced to keep track of which is the most frequent, the second most frequent, and so on.

Figure 2, below, demonstrates the distribution of these different split types across word pairs of different corpora and lengths. For example, 15-letter word pairs from the `linux` corpus (in green) overwhelmingly favour suffix-type splits.

As discussed later in **Results**, we hypothesize that the goblet-like shape seen predominantly in longer words (i.e,., 9+ letters) is due to a widened variety of verbal stems that differ only in their declension, as indicated by a suffix (e.g., participating, participation).

[2]i.e., any naive iterative algorithm.

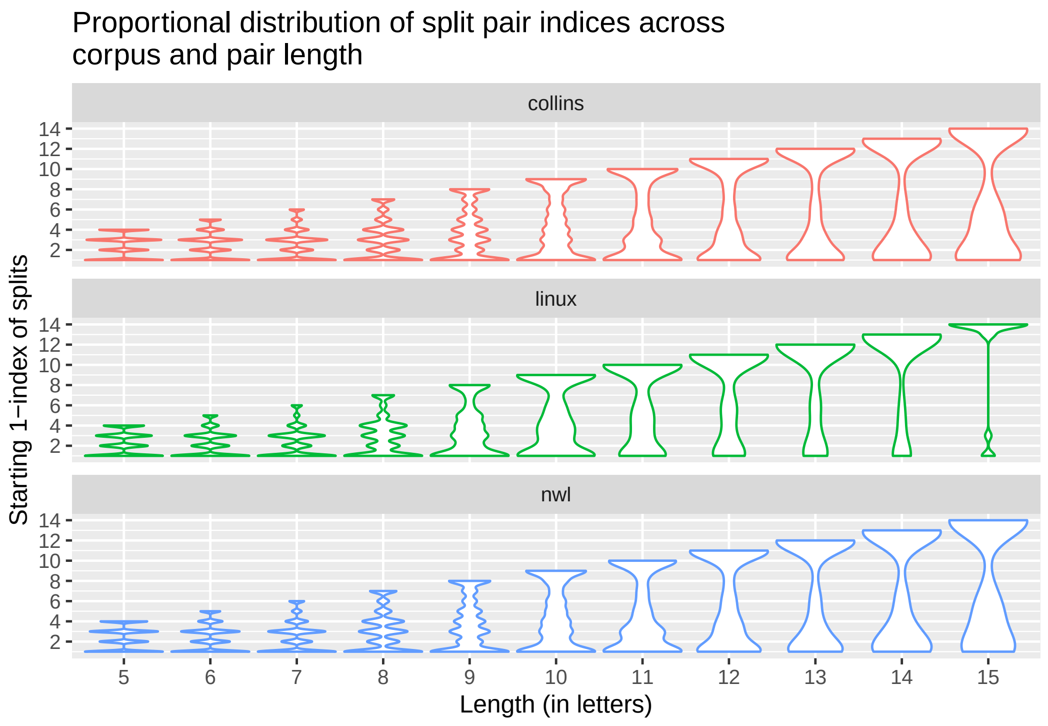


Figure 2: Width at a particular y-value represents the density of splits in word pairs of a given length, starting at that index.

Looking at all consecutive pairings of letters (*2-grams*) in each corpus, we see that 614 out of a possible $26^2 = 676$ pairings are accounted for at least once. Figure 3 below depicts the relative frequency of these pairings, with the white squares representing those pairs that appear nowhere in the corpus (or corpora). Some more infrequent first letters, such as q or z, demonstrate large streaks of unattested pairs. This is in line with certain orthotactic[3] constraints present in the English language and will be discussed in greater detail in **Discussion**.

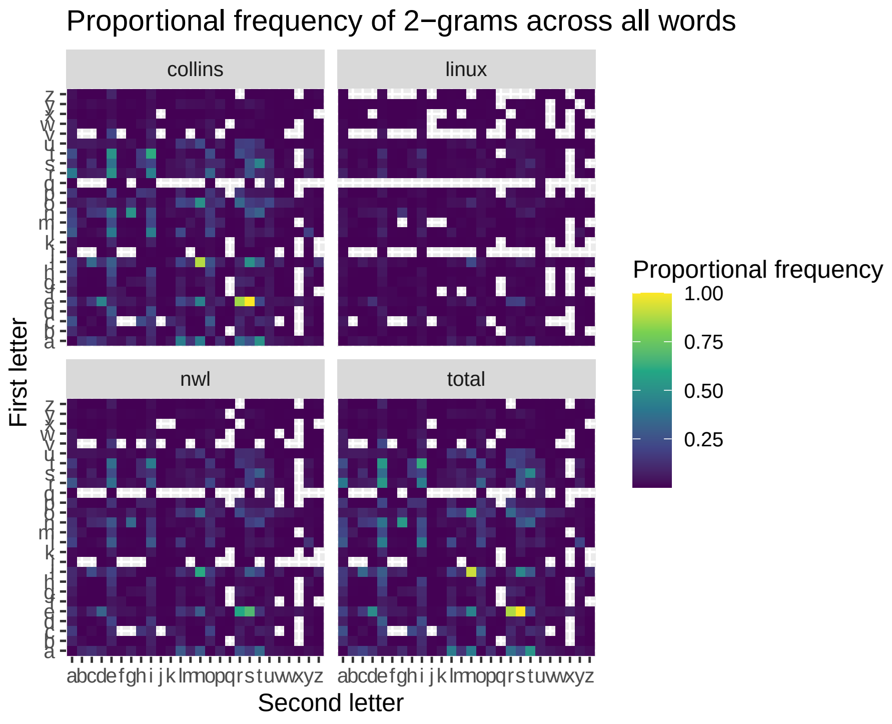


Figure 3: Distributions appear relatively consistent across corpora, except for particularly esoteric combinations: the Linux corpus contains no q's followed by anything other than a u.

Results

Upon encountering such a wide distribution of categorical data (i.e., ordered strings of characters), our first thought was to see if some aspect of the distribution obeyed Zipf's Law (Zipf 1949). In brief, this empirical law states that when observations in a categorical set are ranked by frequency, the second-ranked observation occurs roughly half as often as the first, the third-ranked a third as often, and so on. In mathematical terms, over a frequency-ordered distribution X with ranking variable i,

$$\text{Zipf}(X) \iff X_i \approx \frac{X_1}{i}.$$

[3]Pertaining to the rules and constraints of spelling. E.g., "pwn" violates English orthotactic rules, and thus could never constitute a non-abbreviated English word.

Before examining the splits and other puzzle-specific properties of words, we examined the distribution of one- and two-letter sequences to see if any Zipf-like patterns emerged. Surprisingly, however, the data outperformed Zipf's Law (i.e., contained many more observations than the power law predicts) in these categories. Figure 4 below, which examines the total occurrences of single letters in all valid words, exemplifies this. The black dashed curve represents the expected count of each letter according to Zipf's Law, whereas the total height of each bar represents the actual count. The second-most common letter, s, is expected to occur $539,056/2 = 269,528$ times, or 50% of the total occurrences of e; it instead appears roughly 84% as often, a total of 4.50495×10^5 times. This disparity between Zipfian expectation and reality only increases in higher ranks, save for the universally rare q and j.

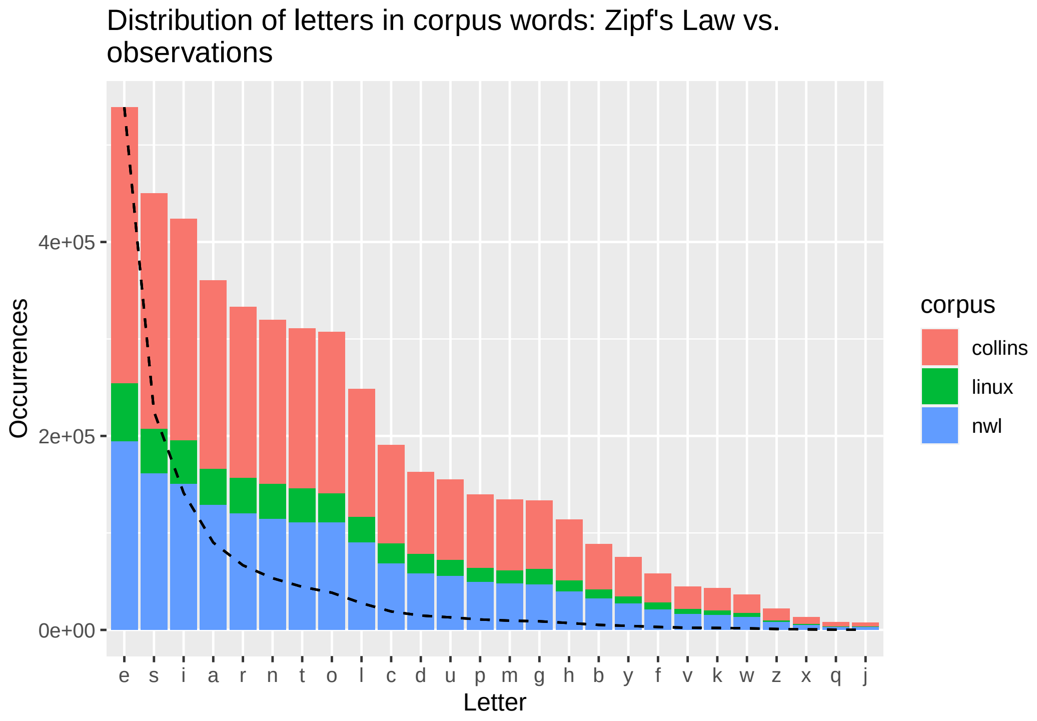


Figure 4: The total distribution of letters across all valid corpus words is consistently greater than the ideal Zipf curve (black) would predict.

Looking at each corpus individually, we can also examine their 2-grams; that is, each consecutive pair of characters that occurs in every valid word. As an example, the word "aardvark" contains the 2-grams "aa", "ar", "rd", "dv", "va", "ar", and "rk". (Note that "ar" occurs twice, and is counted as such.) Figure 5 below again pits the frequency of these 2-grams against the expectation of Zipf's Law, yielding an interesting result. For clarity and visual interest, some of these 2-grams have been included beside their place on the graph, serving as waypoints of increasingly uncommon letter pairings.

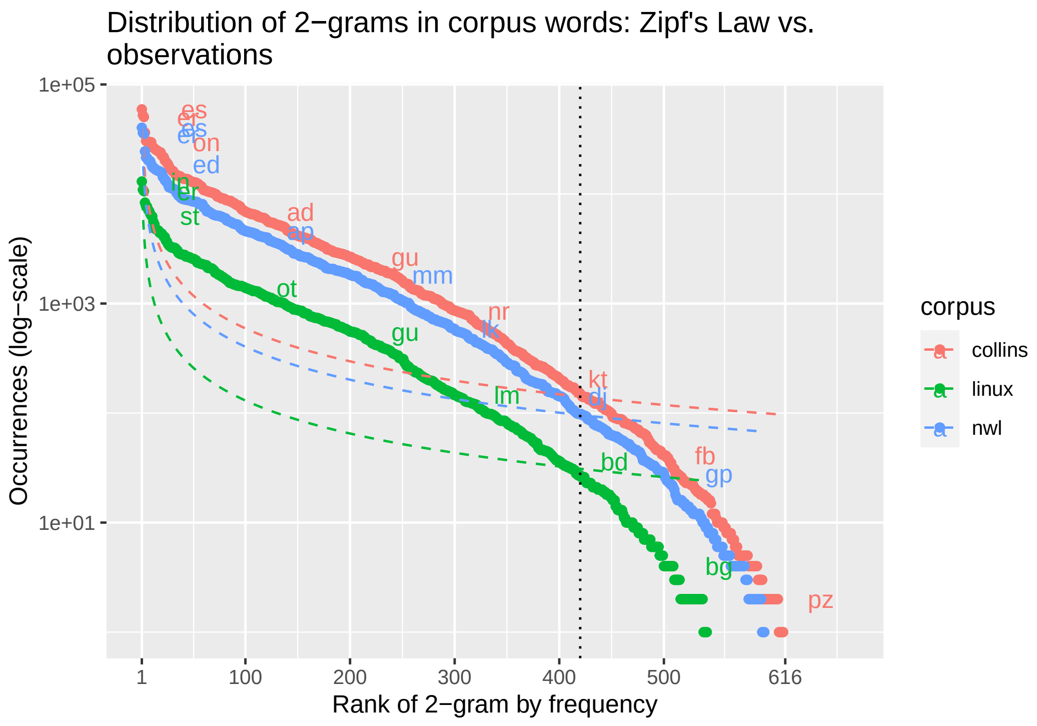


Figure 5: Looking at all 2-grams across the three corpora, distributions of all but the bottom 190 attested instances outperform a Zipf curve.

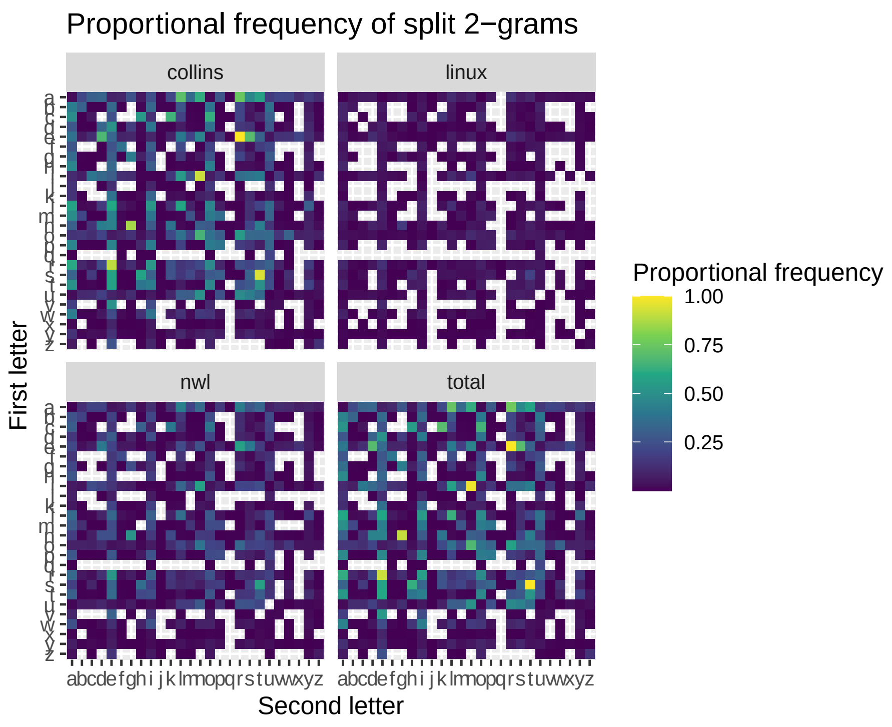
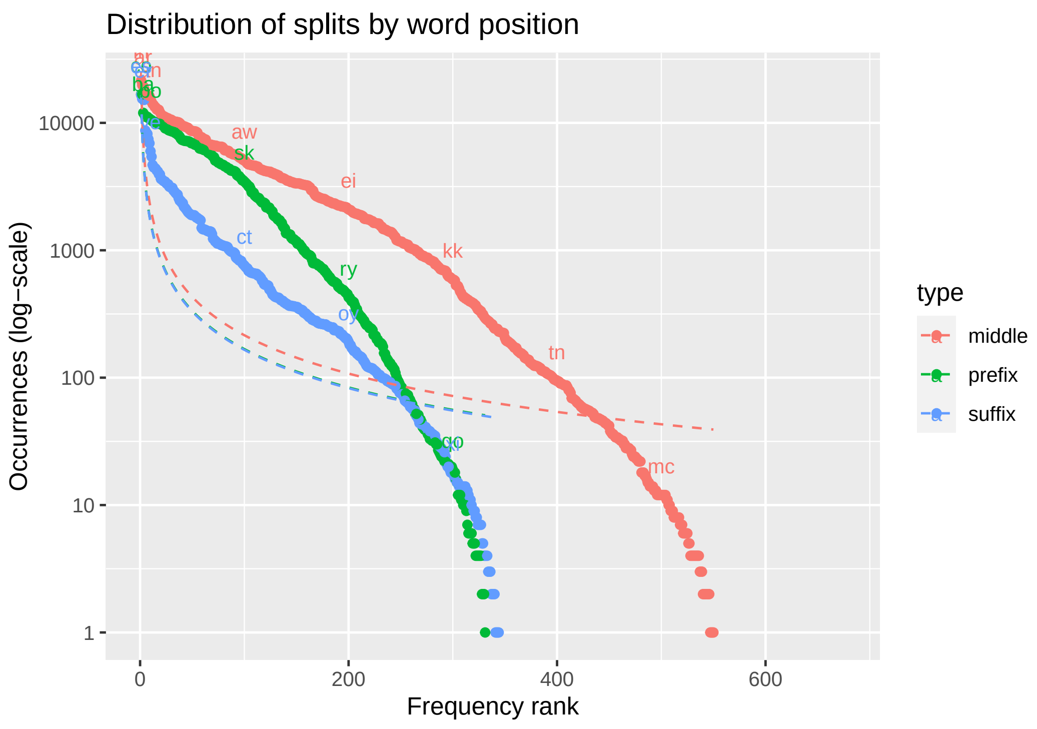


Figure 6: Split pairs are predominated by a few major players: es, ng, re, st, etc.

Table 2: Most common splits by word position.

Rank	prefix	middle	suffix
1	co (16759)	in (21543)	es (16544)
2	re (16740)	ck (19792)	ed (15443)
3	ba (11992)	ar (19450)	er (15269)
4	ca (11639)	ll (18567)	ng (15161)
5	bu (11179)	il (16974)	on (8717)



Discussion

Orthotactics

Phonotactics

Appendix

Link to Python preprocessing script

Found in the GitHub repo.

References

Collins Official Scrabble® Words. 2019. Harper Collins.

NASPA Dictionary Committee. 2020. *NASPA Word List, 2020 Edition.* NASPA.

Piscop, Fred. 2021. *Split Decisions.* Split Decisions.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Xie, Yihui. 2021. *Knitr: A General-Purpose Package for Dynamic Report Generation in R.* https://yihui.org/knitr/.

Zipf, George. 1949. *Human Behavior and the Principle of Least Effort.*