Lexicographic adventures: rhythm and surprises in open source word lists.  
David Dailey and Nathaniel Zeiger  
Slippery Rock University

Table of Contents

Abstract

Introduction

Section 0 – An brief illustration of the sorts of investigations to follow. Words whose letters are alphabetized.

Section I – A selection of resources from which other resources have been made

1. Word lists
2. Thesauruses

Section II – Analysis of characters in lexical resources

Section III – Analysis of sequences of characters

1. Common digrams, trigrams etc.
2. Interchangeable morphemes and polygrams
3. Rhythms of the alphabet

References

**Abstract:**

Over the past 20 years, author Dailey and various SRU students including the second author have built and expanded a variety of lexicographic resources used primarily for the teaching of UNIX shell scripting but also for investigating a variety of linguistic analyses of the English lexicon. This paper briefly summarizes the nature of some of these investigations.

**Introduction:**

Lexicography has a long and distinguished history, with “monolingual Sumerian wordlists in cuneiform writing on clay tablets, ” dating from 3200 BC. [1]. That history has been developed cross-culturally and continually since that time.

At least since the work of Carter Revard in 1967 [2] , the Brown corpus [3] and the beginning of Project Gutenberg in 1971 [4], scholars have expressed interest in obtaining and leveraging computer access to lexicographic resources, particularly those that have entered the public domain. As Michael Hart (the founder of the Gutenberg Project) explained [4]:

The Selection of Project Gutenberg Etexts:   
There are three portions of the Project Gutenberg Library, basically be described as  
Light Literature; such as Alice in Wonderland, [...],  
Heavy Literature; such as the Bible, [...], Shakespeare, etc.  
References; such as Roget's Thesaurus, almanacs, and a set of encyclopedia, dictionaries, etc.

In 1980-82, I worked with Revard and others at the University of Tulsa to compile collections of lexicographic resources, for use in journalistic, anthropological, sociological, linguistic and psychological research. Changes in my academic discipline led me, after that, in other scholarly directions, but since 1999, I’ve often taught a course in Unix shell scripting. I have found that using English word lists as “data” is something that students, who might lack mathematical or actuarial experience, might find to offer a more intuitive dataset than, for example, amortization tables. Accordingly, I’ve tried to keep a set of resources such as English word lists, thesauruses and dictionaries available for students to use to practice their sed, grep and awk skills.

**Section 0 – A brief illustration of the sorts of investigations to follow. Words whose letters are alphabetized.**

In order that the reader might have some idea of the sorts of invesigations that follow, one relatively simple, but at the same time, we think, unusual inquiry is presented.

Q: What do the following words have in common?

dehort  
chintz  
biopsy  
begirt  
almost  
mopsy  
horst  
glory  
gipsy  
ghost  
forty  
first  
filmy  
empty  
dirty  
deity  
deist  
chino  
chimp  
blowy  
bijou  
below  
begot  
begin  
befit  
amort  
ahint  
aglow  
aegis  
adopt  
adept  
adeps  
abort  
abhor

A: The letters of each word are in alphabetical order: For example almost: a<l<m<o<s<t.

In fact, these represent all five and six letter words having this property found in a particular dictionary [4]. The little awk script we used to find them is as follows:

awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS="" FR2009|sort -nk2

(In which FR2009 is the dictionary used [5], see also the description of resource #9 in Section IA, above.) It should be noted for sake of transparency that one “word” found by this script was not included: “‘cept” since it begs the question of whether or not this is a real word, and if so, whether or not the apostrophe is, indeed, alphabetically prior to the lower case alphabetic characters that follow it.

Counting the number of such words, via the commands

awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS="" FR2009|wc –l  
406

and

awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS="" FR2009|sort -nk2|awk '{print $2}'|uniq -c  
 32 1  
 130 2  
 126 3  
 83 4  
 30 5  
 5 6  
we find 406 words (including 32 one letter words, 130 two letter words and so forth).

I should note however, if we use a much larger dictionary, such all words that appear in at least two of the sixteen resources discussed above, then a few extras can be found, including these gems: *horsy, adhort, adipsy, agnosy, befist, begins, behint, beknot, bijoux, cestuy, chimps, chinos, chinoy, deflow, deflux, dehors, deimos, deinos, delors, dhikrs, diluvy, dimpsy, ghosty, deglory, egilops*, and the eight letter *aegilops*, “a genus of Eurasian and North American plants in the grass family” [7].

A natural question, it would seem, is whether or not there are more or fewer words whose letters are in inverse alphabetical order. The answer is consistent across both of these dictionaries: No.

Table 1 shows the number of positively and negatively alphabetical words (as ‘almost’ or ‘sponge’, respectively) for the smaller FRELI dictionary [4] as well as the larger word list TwoOrMore [8].

|  |  |  |
| --- | --- | --- |
| Number of words | FRELI [4] 73735 words | TwoOrMore [7] 406712 words |
| Alphabetical (as a<l<m<o<s<t) | 406 | 2195 |
| InverseAlpha (as s>p>o>n>g>e) | 243 | 1594 |

The words in inverse alphabetical order (i.e., ‘monotonically decreasing’) were found via the following awk script (using, for instance, the Freli words, in FR2009):

awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0,NF }' FS="" FR2009|sort -nk2

Interestingly, as discussed in Section IIC, this predominance of positively alphabetic words over the negatively alphabetic ones persists for all letter lengths greater than one.

This result is, perhaps, counterintuitive at several levels of expectation. Why aren’t there more such words? Why aren’t there longer examples? Why are there more that increase alphabetically than those that decrease? Such questions will be examined in more detail in this paper

**Section I – A selection of resources from which other resources have been made**

**Part A. Word Lists**

In 1980, at the University of Tulsa, I researched the speed with which people could determine when two sequence of letters (words or not) might have the same letters within them. How quickly, the studies asked, might people be able to respond in a reaction time experiment that the words “READ” and “DARE” for example, contain the same four letters? The results were intriguing, with certain categories of permutations and word-nonword pairings being far quicker for most subjects than others. Later analyses suggested strong cross-cultural differences in many of the results. Fundamental to such a study was the collection of anagrams, and for that, access to a “good” list of English words was important. In those days, a list of words for use in word games contained only a few thousand four letter words, and those could be keyed by hand, absent the availability of machine readable resources.

The first large online lexical resources that I became aware of included an earlier version (36,000 words instead of the 2009 version with its 73,000 words) of the FRELI project [4], the 1911 Roget’s thesaurus [8], and the 1913 Webster’s Unabridged Dictionary [9]. The 1913 Webster’s, from the Gutenberg Project [2], was packed with etymologies, complex and inconsistent markup, foreign words, errata, and was too difficult to parse for use in class assignments. The thesaurus, also from Gutenberg, was a bit too limited in raw vocabulary, and was somewhat quirky, owing, perhaps, to Roget’s rather idiosyncratic notions of semantic theory. The vocabulary of the FRELI list, on the other hand, was “believable”. That is, most of the 36,000 words in it, “looked like words”, even when an entry itself might not have been a part of one’s own vocabulary. As will be seen, not all such resources share this property of “face validity” among native speakers of the language. One might ask why we do not use more modern and authoritative resources? The answer: copyright law! Works published prior to 1923 in the United States have entered the public domain, more modern works almost always remain under copyright [10].

Accordingly, one is, for sake of unsponsored academic research and teaching, likely to seek open source, and or, public domain lexical resources.

**A1. Various open source word lists**

Altogether, we’ve been able to locate 16 different lists of English words, each with an open license that allows re-use, typically with attribution of source. In several cases, the work has entered the public domain due to expiration of copyright. These dictionaries vary considerably in age (ranging from the 19th through 21st centures) and degree of curation (some have been carefully curated by lexicographers or dictionary authors, while others have been sampled from large collections of “free range” English text). They also differ in orthographic convention (the handling of hyphenations, apostrophes, capitalization and non-ASCII characters, like é, æ and ñ), and “lexical tolerance” (what we call the degree of rigidity or toleration for such things as slang, misspellings, trademarks, place names, vulgarity, etc.).

Dictionaries are not all the same. In truth, since the days of Samuel Johnson’s first dictionary of the English language [10], the dictionary author’s personality [11] has influenced the final product, and, as is obvious from a cursory look at any “old” dictionary [10, 13], the English language changes over time. Many dictionaries come from a particular philosophical perspective and have more or less tolerance for the speech of the masses. For example, Webster was more prudish than Samuel Johnson, though he still came under some criticism for including too many vulgar words in his dictionary. In response he wrote

...one thing must be acknowledged by any man who will inspect the various dictionaries in theEnglish language, that if any portion of such words are inadmissable, Johnson has transgressed the rules of lexicography beyond any other compiler, for his work contains more of the lowest of all vulgar words than any other now extant ... Any person who will have the patience and candor to compare my dictionary with others will find that there is not a vocabulary of the English language extant more free from local, vulgar, and obscene words as mine.

By the time of the publication of the 1973 edition of the Webster’s New Collegiate Dictionary [13] all of the seven words that George Carlin in his famous 1972 monologue [12] on the subject said could not be used on television, were indeed “in the dictionary.”

While certain lexical resources (e.g., the 1919 Webster’s Collegiate Dictionary) [13] are now available online, the text for such has been converted to digital text through OCR, rendering many of those resources very difficult to use. For example, here are two entries (for consonantly and consort, respectively) from the version available from The Internet Archive.

OOn'SO-nant-ly, adv. In consonance ; in accord.

eon-sort' (kon-s6rt'), v. i. To unite ; associate ; also, to ac-

cord ; agree. — v. t. To escort or attend ; accompany. 06s.

Cleaning up this sort of materials for further use would require a major effort. Fortunately, the Gutenberg Project has often crowd-sourced the human effort to provide reliable machine readable versions of such lexical resources. Many of the resources that we’ve used come from there.

Another very noteworthy resource is the Wiktionary (located at <https://www.wiktionary.org/> ). It claims over 5 million entries for English, and is an open source, collaborative project. Unfortunately, as of this writing, though the words in the database are searchable, we have been unable to find a way to download the lexical data to be able to query it repeatedly or to compare and contrast its data with the other resources we have made use of.

Here are the particular resources we’ve been able to use. By “use” we mean to find open access versions of, which are of sufficiently consistent encoding that they may be “consolidated” and compared to see the degree to which these resources overlap and differ. In some of the cases below, e.g., #10, AllenW, and #3 WebsUni, the raw dictionary itself was downloaded as a text file and then converted first to a word frequency tabulation and then to a listing of unique words used in the resource. So when I reference 138,000 words in the Webster resource, this doesn’t mean that all of those words had definitions within the dictionary, since some of those terms may have been used in the definitions of other words.

Here are the particular resources we’ve been able to use, with brief descriptions.

1. 74550com.mon from the Moby Project [15, 16].   
74,550 common dictionary words. A list of words in common with two or more published dictionaries.  
  
2. 354984si.ngl from Moby Project [15, 16].   
354,984 single words. Over 354,000 single words, excluding proper names, acronyms, or compound words and phrases. This list does not exclude archaic words or significant variant spellings.  
  
3. WebsUni -- From Webster's Revised Unabridged Dictionary, 1913.  
138,900 words. In the public domain. (See https://en.wikipedia.org/wiki/Webster's\_Dictionary#1913\_edition)  
The version we have used (see http://granite.sru.edu/~ddailey/cgi/readwebster?wild)  
comes from the OPTED project: http://www.mso.anu.edu.au/~ralph/OPTED/index.html  
  
4. Engwords -- https://invokeit.wordpress.com/frequency-word-lists/  
456,631 words English Word Frequency list. Downloaded Fall 2016 from Hermit Dave. Licensed under Creative Commons – Attribution / ShareAlike 3.0. These word frequency lists were generated through scripts which excerpt data from the Open (movies) Subtitles Database at http://opus.lingfil.uu.se/OpenSubtitles2012.php . See https://invokeit.wordpress.com/about/ for further explanation   
  
5. 113809of.fic from Moby Project [15, 16].  
113,809 words. A list of words permitted in crossword games such as Scrabble(tm). Compatible with the first edition of the Official Scrabble Players Dictionary(tm).   
  
6.USDW -- /usr/share/dict/words  
479,828 words. The historic UNIX/Linux spell checker [17,18]. The versions in current Linux distributions seem to be largely based on the SCOWL project [19].   
  
7. Awords -- Academic Words from Corpus of Contemporary American English [20].   
18559 words. The COCA project at Brigham Young University provides some information (like this vocabulary of words derived from academic journals) free of charge. The academic words represent a source of reliable entries though under-represented in word frequency analyses stemming from other sources.  
  
8. BNCwords -- Words from the British National Corpus.  
131237 words The British National Corpus (BNC) [21] is a “100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English, from the late twentieth century.” The subset used here: BNCwords is a subset of the British National Corpus provided by Adam Kilgariff as described at http://www.kilgarriff.co.uk/bnc-readme.html.   
  
9. FRELI -- http://www.nkuitse.com/freli/  
73177 words. "FRELI (the Free Repository of English Lexical Information) ..." This is release 20090227 of FRELI (the Free Repository of English Lexical Information), a freely redistributable list of English words with associated information (parts of speech, alternate spellings, etc.). Creative Commons Attribution License, version 2.0.   
  
10. AllenW -- from Allen's Synonmyms and Antonyms by F. Sturges Allen.  
56077 words. 1920 Published by Harper and Bros., hence in the public domain. See https://archive.org/details/allenssynonymsan00alle. Downloaded from the Gutenberg Project.  
  
11. SouleW -- from A Dictionary Of English Synonymes And Synonymous Or Parallel  
Expressions Designed As A Practical Guide To Aptness And Variety Of Phraseology By Richard Soule Boston: 1871.  
27417 words. In the public domain and downloaded from the Gutenberg Project.  
  
12. FallowsW -- from A Complete Dictionary Of Synonyms and Antonyms Or Synonyms  
and Words of Opposite Meaning. 1898 by Rev. Samuel Fallows, A.M., B.D.  
19474 words. Public domain and accessed from the Gutenberg Project as digitized and reset by Steve Wood 2016.  
  
13. FernW -- English Synonyms and Antonyms With Notes on the Correct Use of Prepositions By James C. Fernald, L.H.D. Nineteenth edition, Funk & Wagnalls Company New york and London, 1896  
16132 words. In the public domain and accessed from the Gutenberg project.  
  
14. PutW -- Putnam's Word Book  
Putnam's Word Book. A Practical Aid in Expressing Ideas through the Use of an Exact and Varied Vocabulary (Under the title Synonyms, Antonyms, and Associated Words). Louis A. Flemming. Copyright, 1913 By G. P. Putnam's Sons.  
29732 words. In the public domain and accessed from the Gutenberg project.  
  
15. 2Words 2of12full.txt  
48564 words. From the SCOWL project [19]. Specifically, it comes from the 12Dicts package of Alan Beale.  
The file 2of12full.txt contains the all words appearing in more than than one of Alan Beale's source dictionaries.  
  
16. W2Words BYU COCA ngrams (2 words).   
68784 words. A word list derived from the 2word ngrams from COCA. The COCA project [20] makes available for free, selected ngram data presenting the frequency of cooccurrence of pairs of English words.

As can be observed, these resources are quite heterogeneous, concerning their dates of origin as well as the methodologies by which they were assembled or harvested. Some are likely quite well-curated, with the dictionary makers having painstakingly deliberated over the entries. However, with some of the older resources, and carefully curated resources, there are OCR errors. With some of the more modern resources, the nature of harvesting-based methodology has allowed certain amounts of noise to enter the data – such is likely to be the case with gathering data from garden variety users of the language.

**A2. Various idiosyncracies associated with these resources**

At first glance, it might seem straightforward to simply form the set theoretic union of the sixteen word lists described in the earlier section. In unix/linux, the comm command is built precisely to examine intersections and unions of the lines of two files. [23]

In the earlier version of the FRELI project [5], within the first screenful of words is the following:

abb&eacute;

This particular resource uses “HTML character entities” like “&eacute;” to represent the Unicode character: é. That is, by “abb&eacute; “is meant “abbé” . Some of the files followed this convention while others used the actual Unicode characters. Accordingly, a script needed to be written convert HTML entities to their Unicode equivalents.

Initially, resources were scanned for such characters, using a simple grep. Displaying the actual character next to the encoding required a bit of tedium for the translation. (The program UniTrans, described below).

|  |  |
| --- | --- |
| Finding and replacing HTML entities used in resource #3 WebsUni | |
| Command: paste <(grep -o "&[^&;]\*;" WebsWords|sort|uniq -c) <(for i in `grep -o "&[^&;]\*;" WebsWords|sort|uniq`; do echo $i `echo $i|./UniTrans`; done)|awk '{print $1, $2, $4}'  Output: | |
| 7 &aacute; á  6 &acirc; â  528 &aelig; æ  5 &agrave; à  1 &atilde; ã  8 &auml; ä  14 &ccedil; ç  232 &eacute; é  18 &ecirc; ê  41 &egrave; è  174 &euml; ë | 1 &icirc; î  14 &iuml; ï  15 &ntilde; ñ  1 &oacute; ó  7 &ocirc; ô  179 &oelig; œ  152 &ouml; ö  3 &ucirc; û  1 &ugrave; ù  12 &uuml; ü |

As we could find nothing in the standard Linux distribution to do this , the program UniTrans is merely a chain of sed substitution commands:

sed '  
s/&aacute;/á/g;  
s/&acirc;/â/g;  
s/&aelig;/æ/g;  
s/&AElig;/Æ/g;  
s/&agrave;/à/g;  
s/&atilde;/ã/g;  
s/&auml;/ä/g;  
s/&ccedil;/ç/g;  
s/&Ccedil;/Ç/g;  
s/&eacute;/é/g;  
s/&Eacute;/É/g;  
s/&ecirc;/ê/g;  
s/&egrave;/è/g;  
s/&euml;/ë/g;  
s/&icirc;/î/g;  
s/&iuml;/ï/g;  
s/&ntilde;/ñ/g;  
s/&oacute;/ó/g;  
s/&ocirc;/ô/g;  
s/&oelig;/œ/g;  
s/&OElig;/Œ/g;  
s/&ouml;/ö/g;  
s/&ucirc;/û/g;  
s/&ugrave;/ù/g;  
s/&uuml;/ü/g;'

Another of the problems is the handling of proper names. Some don’t include them; others do, but signal them with initial capital letters. Still others include many word tokens twice: once capitalized and once not (for example “the” and “The” both being included in the word list). Our general approach to this has been to convert all characters (including such things as ‘Æ’ and ‘æ’) to lowercase. It would require manual intervention to discriminate between proper names that had been converted to lowercase by a particular author from “regular” words, so, no attempt to “cleanse” the actual words themselves was made.

Additionally, it was found that some resources (like the British National Corpus) had multiple entries for words, based on the different sense, or part of speech that the word (like “left” as either an adjective or a verb) might have. Hence, sorting each of the files and running it through *uniq* (which removes duplicates) was essential.

Some of the files came with carriage return + new line sequences between the words, while others used the standard Unix convetion of just the “\n” as the record delimiter. In order to merge files, it was important that this be standardized! Some, like McomX and WebsUni allow certain multi-word entries (like *ad hoc*) while others have only one “word” per lexical entry.

Finally, after the basic ground rules of the files (and encodings of the files) had been standardized, it was time to compare these datasets. First, we present a cursory view of the sizes of each resource:

$ wc McomX MSinX WebsUni EngWords MoffX USDW Awords BNCwords F09u AllenW SouleW FallowsW FernW PutW 2Words W2Words  
   74550    89925   730682 McomX  
  354983   354983  3712683 MSinX  
   98532   101505   962913 WebsUni  
  456631   456631  4034760 EngWords  
  113809   113809  1016714 MoffX  
  479828   479828  4953680 USDW  
   18559    18559   169241 Awords  
  131237   131236  1163970 BNCwords  
   73735    73735   759817 F09u  
   56077    56077   485918 AllenW  
   27417    27417   251972 SouleW  
   19474    19474   189341 FallowsW  
   16132    16132   140345 FernW  
   29732    29732   274018 PutW  
   48564    48564   445085 2Words  
   68784    68784   603383 W2Words  
2068044  2086391 19894522 total

Next, a partial attempt to view the redundancy and overlap of pairs of these resources was made. For two resources, like WebsUni and BNCwords, the *comm* command can be used to determine the size of the intersection of the two word lists.

Specifically, in this case

$ comm -12 WebsUni BNCwords|wc

reveals that only 29823 words are in common to the 98532 words in WebsUni and the 131237 words in BNCwords. That number is entered into the table in the lower triangular portion of the matrix where the two meet. Where WebsUni intersects itself, there is listed the total size of the resource, for easy reference.

The comm command can also be used to form differences between two sets (WebsUni – BNCWords or BNCWords - WebsUni ). The smaller of these two numbers is represented in the upper diagonal portion of the matrix. Not all of these analyses were performed, since the overlap data (containing intersections) gave the bulk of the information sought about the degree of redundancy between the resources.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | McomX | MSinX | WebsUni | EngWords | MoffX | USDW | Awords | BNCwords | F09u | AllenW | SouleW | FallowsW | FernW | PutW | 2Words | W2Words |
| McomX | 74550 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MSinX | 46940 | 354983 |  |  | 2589 | 3662 |  |  |  |  |  |  |  |  |  |  |
| WebsUni | 36946 | 81824 | 98532 |  |  |  |  | 68709 |  |  |  |  |  |  |  |  |
| EngWords | 33590 | 98423 | 38691 | 456631 |  |  |  |  |  |  |  |  |  |  |  |  |
| MoffX | 33977 | 111220 |  | 66987 | 113809 |  |  |  |  |  |  |  |  |  |  |  |
| USDW |  | 351321 | 84285 | 99411 |  | 479828 |  |  |  |  |  |  |  |  |  |  |
| Awords |  |  |  | 16975 |  |  | 18559 |  |  |  |  |  |  |  |  |  |
| BNCwords | 28325 | 65487 | 29823 | 78863 | 49668 | 69525 | 17521 | 131237 |  |  |  |  |  |  |  |  |
| F09u |  |  | 40122 | 39463 |  | 63988 | 15990 | 33083 | 73735 |  |  |  |  |  |  |  |
| AllenW | 25570 | 38024 | 30149 | 28349 | 40147 | 40147 | 12383 | 23525 | 26224 | 56077 |  |  |  |  |  |  |
| SouleW | 17541 |  | 20309 |  | 20221 | 25266 |  |  |  | 18348 | 27417 |  |  |  |  |  |
| FallowsW | 12670 |  | 14089 |  |  | 17152 |  |  |  | 13683 | 13668 | 19474 |  |  |  |  |
| FernW |  |  |  |  |  | 13744 |  |  | 9595 | 10019 | 10103 | 8474 | 16132 |  |  |  |
| PutW |  |  |  |  |  | 28457 |  |  | 21963 | 19912 |  |  |  | 29732 |  |  |
| 2Words |  | 40627 |  | 36356 | 32599 | 47058 |  |  | 33846 | 22608 | 18003 |  |  | 19556 | 48564 |  |
| W2Words | 22614 | 46694 |  | 57018 |  | 48685 |  |  | 25457 |  |  |  |  |  | 26608 | 68784 |

As we compared and contrasted these various sets of words, several of the idiosyncrasies of the word lists became apparent. A cursory analysis of the UNIX words, USDW, for example reveals something that all who have looked at that resource probably realize: there are a lot of things that don’t look like words, for example:

$ head -12 USDW|tail -2  
2,4,5-t  
2,4-d

Given that the file’s history is not just for use in spell-checking but also for validating that passwords are “secure”, it makes some sense that many things in it would not be conventional words. While a typical unabridged dictionary of English might have 150,000 words, the nearly half a million entries in USDW are certain to have many oddities. An analysis of the length of words in USDW is revealing containing words of length 29, 30, 31 and even 45 including for example: ‘dichlorodiphenyltrichloroethane’, ‘half-embracinghalf-embracingly’ and ‘pneumonoultramicroscopicsilicovolcanokoniosis’ [24]

An analysis of the length of words in USDW is revealing:

$ cat USDW|awk '{print NF}' FS=""|sort -n|uniq -c

53 1  
 1271 2  
 6221 3  
 13208 4  
 25104 5  
 41699 6  
 53944 7  
 62334 8  
 62615 9  
 54667 10  
 46510 11  
 37583 12  
 27976 13  
 19326 14  
 12160 15  
 7137 16  
 4014 17  
 2010 18  
 1055 19  
 508 20  
 240 21  
 103 22  
 50 23  
 19 24  
 9 25  
 2 26  
 3 27  
 2 28  
 2 29  
 1 30  
 1 31  
 1 45

This shows word lengths that are generally typical of English words (varying between 2 and 15 letters, but there there are some really long words too:

$ cat USDW|awk 'NF> 25 {print $0}' FS=""

antidisestablishmentarianism  
cyclotrimethylenetrinitramine  
dichlorodiphenyltrichloroethane  
electroencephalographically  
half-embracinghalf-embracingly  
hydroxydehydrocorticosterone  
hydroxydesoxycorticosterone  
Mentor-on-the-Lake-Village  
microspectrophotometrically  
pneumonoultramicroscopicsilicovolcanoconiosis  
straight-from-the-shoulder  
trinitrophenylmethylnitramine

“Pneumonoultramicroscopicsilicovolcanoconiosis”, by the way, is “a word invented by the president of the National Puzzlers' League as a synonym for the disease known as silicosis. It is the longest word in the English language published in a dictionary, the Oxford English Dictionary, which defines it as "an artificial long word said to mean a a lung disease caused by inhaling very fine ash and sand dust. “ [24]

The BNCwords list (from Oxford University, of all places) contains such entries as

$0.0090497  
00.00z  
+0.068  
0.1&ins  
031–469  
0.5°c  
⅛pt  
£100,000-a-year  
a.agassizii  
adrichem-boogaert  
aef-1&agr

The BYU COCA list had some oddities as well. The list contained about 6000 words including ‘backsplash’ and ‘baby-boomer’ that were not in the 15 other dictionaries. But looking deeper, we found a curiosity: eleven words that contain ‘@’ as a letter. Each of these appeared to be the email address of a journalist (e.g., ‘talk@npr.org’): rather an odd choice for inclusion in a list of words.

While one can certainly imagine rules to assist one in culling these various resources to reduce some of the pandemonium, human curation is ultimately the only key to properly cleansing the lists of the obvious nonwords. The problem of course, is that one person’s obvious nonword may be perfectly legitimate to another. Ultimately, by using the approach of 12Dicts package of Alan Beale (as in resource # 15, 2Words ), we may view not just the word’s frequency of usage within the language, but the number of lexical resources in which it appears as a good indicator of the word’s validity. We will see in Section II still more reasons (probably stemming mainly from foreign words used in English texts) to take these data with a bit of suspicion. Nonetheless, if a word appears in 2 or more of the dictionaries, then it is far less likely, it would seem, to be “noise” of one sort or another.

Next, the union of all sixteen dictionaries was created:

$ cat McomX MSinX WebsUni EngWords MoffX USDW Awords BNCwords F09u AllenW SouleW FallowsW FernW PutW 2Words W2Words|sort|uniq -c >WordsInManyPlaces  
  
$ wc WordsInManyPlaces  
  946943  1911581 16794870 WordsInManyPlaces

When this process was completed WordsInManyPlaces contains about 1 million distinct “words” as well as a number representing the number of resources in which that word occurs. The latter data is particularly interesting, since it represents the “wordiness” of a word: that is how many of these resources actually attest to this lexical entry as being an actual word.

**A3. An amalgamated approach to a “meta-dictionary”**

By combining the talents and efforts (both manual and digital) of many different lexicographers, one can perhaps arrive at a meta-resource that overcomes some of the limitations of each. While a simple “union” of word lists, might compound the errors of each, knowing the degree of unanimity associated with an entry is perhaps a better indicator of its validity than mere usage. Many lexicographers have made lists of frequenty misused words , implying that not all uses are considered valid, and that some “misuses” are indeed “frequent.” People rely on dictionaries to be “authoritative.” There seems to be a historical unwillingness to allow the concept of a “word” to reflect momentary whims. At the same time, language changes. While few people over the age of forty know the meaning of “parkour” (an extreme sport), almost all of my students (based on in-class surveys) do. At the same time, I have observed that few of my students know the meaning of the word “platen” (a part of a typewriter) .

While the actual speakers of a language might be guilty of all manner of slang, informality and even intentional innovation, the lexicographer is generally interested more in studying “the language” than people’s abuse of it (intentional or otherwise). Nonetheless one can posit that “abuse” is often one of the driving forces which compels the changes in language that make it so very interesting.

We have provided at <http://cs.sru.edu/~ddailey/cgi/Wotsa?15> is a meta resource that, for a given number 0<n<17 returns a random sample of words that are in precisely n of the sixteen word lists described herein. If one seeks words that are unanimously accepted, then “16” can be chosen as the value of theparameter. If one’s threshold for lexicographic authenticity is considerably more relaxed, then one might choose 1, and see samples of the half million words that are in only one of the dictionaries.

Here are some data showing the number of words in n dictionaries as a function of n:

|  |
| --- |
| $  awk '{print $1}' WordsInManyPlaces |sort -n|uniq -c |
| 539604 1  192198 2   93214 3   34912 4   20992 5   18011 6    9651 7    6755 8    5407 9    4495 10    3734 11    3373 12    3038 13    3088 14    3294 15    5177 16 |

Observe that a) the number of words in only one word list is larger than the size of any one of the word lists; b) the number of words common to all 16 is only about 5000. There is, from these resources, substantially less inter-rater reliability concerning our lexicon than one might imagine.

Following, for sake of illustration, such that the reader might get a sense of just what sorts of words belong to each of these sixteen categories of “wordiness,” are random examples of each.

$ for i in `seq 1 16`; do echo $i ; shuf -n 10 <(grep ^[[:space:]]\*$i[[:space:]] WordsInManyPlaces); echo -----------------------------; done

1 quick-speaking  
      1 shwei  
      1 patchway  
      1 tribout  
      1 esaka  
      1 GRI  
      1 ferch  
      1 surgirá  
      1 sapajo  
      1 twojego  
-----------------------------  
      2 unprotect  
      2 sheaveless  
      2 scurfer  
      2 disconnecter  
      2 ophthalmetrical  
      2 sociol.  
      2 coercement  
      2 mastroianni  
      2 thornlessness  
      2 prigdom  
-----------------------------  
      3 lanosities  
      3 irrupts  
      3 demotions  
      3 Naraka  
      3 inglesa  
      3 moonet  
      3 deregulations  
      3 dulcimers  
      3 disconnector  
      3 bestraught  
-----------------------------  
      4 first-aid  
      4 moulins  
      4 serenatas  
      4 ries  
      4 citrons  
      4 defoliator  
      4 quadrilocular  
      4 coziest  
      4 cyanamide  
      4 immortalizing  
-----------------------------  
      5 welly  
      5 chromed  
      5 merozoite  
      5 phycology  
      5 haling  
      5 ukraine  
      5 bimodal  
      5 lamas  
      5 spindleshanks  
      5 porkers  
-----------------------------  
      6 trundled  
      6 hundredths  
      6 bespangled  
      6 challenges  
      6 autosuggestion  
      6 shirts  
      6 decennium  
      6 proteus  
      6 metachronism  
      6 battlefields  
-----------------------------  
      7 landscapist  
      7 quintal  
      7 captiously  
      7 gastrocnemius  
      7 heathery  
      7 tantalus  
      7 lowe  
      7 tonsured  
      7 compounds  
      7 speculations  
-----------------------------  
      8 forewarning  
      8 poppa  
      8 scowling  
      8 oxygenation  
      8 carney  
      8 vicariate  
      8 cochleate  
      8 presser  
      8 hyperspace  
      8 mouthwash  
-----------------------------  
      9 lefty  
      9 sideshow  
      9 wheaten  
      9 nosed  
      9 tyne  
      9 introducing  
      9 genealogical  
      9 poi  
      9 cess  
      9 summing  
-----------------------------  
     10 diatonic  
     10 incase  
     10 adulterated  
     10 chimp  
     10 dyslexia  
     10 phoneme  
     10 emplacement  
     10 titillation  
     10 sidestep  
     10 drifter  
-----------------------------  
     11 lands  
     11 endoscope  
     11 devilry  
     11 whoa  
     11 scupper  
     11 caduceus  
     11 milkweed  
     11 rotor  
     11 guesswork  
     11 cafeteria  
-----------------------------  
     12 imbroglio  
     12 archery  
     12 skyward  
     12 shamefaced  
     12 thence  
     12 trophic  
     12 prosody  
     12 whiskey  
     12 timekeeper  
     12 starched  
-----------------------------  
     13 matronly  
     13 consciously  
     13 biscuit  
     13 groundless  
     13 limbo  
     13 stairs  
     13 cubicle  
     13 merger  
     13 assimilation  
     13 orgasm  
-----------------------------  
     14 daze  
     14 tobacco  
     14 dissatisfied  
     14 patriotic  
     14 adjoin  
     14 mongrel  
     14 invariable  
     14 nightfall  
     14 totter  
     14 brat  
-----------------------------  
     15 crusty  
     15 evacuate  
     15 wane  
     15 nutritious  
     15 hitch  
     15 recourse  
     15 confirmed  
     15 inducement  
     15 tavern  
     15 nasty  
-----------------------------  
     16 garb  
     16 impulsive  
     16 sauce  
     16 obey  
     16 actuality  
     16 physical  
     16 sever  
     16 question  
     16 communicative  
     16 obliterate  
-----------------------------

As a quasi-practical example of the use of these data, in 2015 Dailey created a game using some of them. I wanted to make a word seach game, in which words were randomly selected, and from which people could find collections of words by traversing letters in geographic proximity to one another. To guarantee that the game could be solved, the letters would be chosen from actual words sampled randomly. But at the same time, players could select letters in any order, so that anagrams of words could be recognized by the program. The reader can experiment with the game here: <http://cs.sru.edu/%7Eddailey/slidewords.htm> . The key to such things seems to be that players (self included) are displeased when they find words, but the software fails to recognize that it is a word. At the same time, one of my colleagues said he liked the game and would be interested if his grandchildren could play. That implied to me that I should take some care not to include the standard obscenities in the game’s vocabulary. The problem I encountered was that if I set the threshold high enough to exclude obvious obscenities, then it was also so high that many well-known and perfectly legitimate seeming words would be excluded. I spent some time in the first few weeks of testing the program, manually adding words when I found they were not present! There seems to be no substitute for hand-curation of these resources.

**Section IB Thesauruses**

We won’t go into great detail in this description except to say that we’ve done far more work than we can summarize in this venue, and that our plans for moving further are extensive.

First, it is perhaps important to realize that Roget, upon whose work many of the lexical resources of the web are based, had his own somewhat idiosyncratic theories of semantics. Words, according to Roget, could be classified taxonomically. They belonged in categories. And Roget’s thesaurus, accordingly, tending to list for any given word, hundreds of “synonyms.” All mammals, for example are found in the same category, and are therefore, seen as synonyms of one another. In my work in the 1980’s on synonymy I wrote about the “Rogetian distance” between two words based on the graphic theoretical minimum number of links needed to be traversed in a synomy graph. Alas, Roget’s actual thesaurus was uniquely ill-suited for use in such an exercise because of the great bushiness of the graph. Nodes in that graph simply have too many neighbors. On the other hand, the other more up-to-date thesauruses are all, still, under copyright.

Hence around 2000, I came up with a “bootstrapped thesaurus” which slightly improved upon Roget’s by taking the Grady Ward thesaurus [15] which itself was largely derived (so it appears, though a statement of Ward’s actual methodology does not seem to exist) from Roget’s. The bushiness and the mammals sort of show the family resemblance. My idea was fairly simple: the degree to which two words share a preponderence of the same synonyms gives us a better reflection on the strength of their synonymy. So in the resource available since 2000 at <http://granite.sru.edu/~ddailey/cgi/hyphens?wild>. In the “relative” section of that web site, is where one can see this approach at work. When one enters a word like “wild” one sees the best fourteen synonyms, where for the word wild, each of the words which cooccur with wild somewhere in the relational databse, is itself looked up, and we see the degree to which those words continually reappear as neighbors of the neighbors of “wild.” Some of this work was presented in a presentation by Shirk and Dailey , 2011. That approach has recently been extended through a fairly complex graph theoretic consideration in which the outbound and inbound connections for given entries are compared so as to diversify the relatives of a given node. The results of that approach can be seen in the section of <http://granite.sru.edu/~ddailey/cgi/hyphens?wild> labeled “new approach.”

However, with the discovery of new lexical resources (new thesauruses coming into the Gutenberg project, namely resources 10,11,12,13 and 14) the hope and eventual plan is to consolidate the strength of the connection between word pairs by considering in addition to the measures above, with the consensual data offered by multiple authors. Refining a core vocabulary suitable for the creation of such a thesaurus is a preliminary step, but fortunately the work presented in section IA largely accomplishes this particular goal.

**Section IIIC Rhythms**

**Rhythms of the language**

Alphabets, syllabaries, idiographies – the choice of a writing system may be influenced by a language’s cadence.

The choice of how a language invents a Pig Latin may as well.

Consider the following:

forty 5  
ghost 5  
gipsy 5  
glory 5  
mopsy 5  
almost 6  
begirt 6  
biopsy 6  
chintz 6  
dehort 6

On probabiliities of monotonic (and other) letter sequences

Motivation: there are more words whose letters are in alphabetical order than whose letters are in inverse alphabetical order:

#(alpha order)  
$ cat $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS=""|sort -nk2|wc

212 424 1362

#(inverse alpha order)  
$ cat $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0,NF }' FS=""|sort -nk2|wc

145 290 914

Examples:

$ cat $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS=""|sort -nk2|tail  
forty 5  
ghost 5  
gipsy 5  
glory 5  
mopsy 5  
almost 6  
begirt 6  
biopsy 6  
chintz 6  
dehort 6  
$ cat $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0,NF }' FS=""|sort -nk2|tail  
polka 5  
solid 5  
sonic 5  
spoke 5  
theca 5  
tonic 5  
unfed 5  
wrong 5  
sponge 6  
vomica 6

This observation led to an investigation of “lexical letter rhythms,” as well as curiosity about   
a) whether the above points to some “preference” of monotonically increasing sequences, or simply to the possibility that more English words begin with letters early in the alphabet, hence making increasing sequences more probably

b) whether the rhythms of monotonicity in letter sequences favor certain patterns more than others

c) the extent to which all of this can be explained by pure randomness.

Let [∈](https://en.wikipedia.org/wiki/%E2%88%88) {a..z}\* with ||=2 and  =a1a2. (In English, this just means let the symbol alpha refer to a string of two lowercase letters (a1 and a2) from the English alphabet.) Let us write a1 < a2 to mean that a1 is alphabetically prior to a2 .

If  is chosen at random from {a..z}\*, then P(a1 = a2) = 1/26 and P(a1 < a2)= ½ (25/26) [≈](https://en.wikipedia.org/wiki/%E2%89%88) .48 .

In actuality, of the 43 two letter words in w$:

$ egrep ^[a-z]{2}$ $w

ah  
am  
an  
as  
at  
ax  
ay  
be  
bo  
by  
do  
em  
en  
ex  
fa  
go  
ha  
he  
id  
if  
in  
is  
it  
la  
lo  
me  
mi  
my  
no  
of  
oh  
on  
or  
os  
ox  
pi  
re  
so  
to  
up  
us  
we  
ye

$ egrep ^[a-z]{2}$ $w|wc  
 43 43 129  
24 of them have a1 < a2, while the other 19 have a1 > a2 . This is not likely outside the expectations of chance.

For longer words, though, the situation is more complex. Let’s consider three letter sequences, both English words and nonwords.

For arbitrary letter sequences , [∈](https://en.wikipedia.org/wiki/%E2%88%88) {a..z}\* with ||=2 and  =a1a2 … an ,we call a letter sequence monotonic increasing if ai < aj for all i and j less than n+1. It is monotonic nondecreasing if [∀](https://en.wikipedia.org/wiki/%E2%88%80) i,j ai [≤](https://en.wikipedia.org/wiki/Inequality_(mathematics)) aj .

Examples:

 =abc is monotonic increasing, but is not a word.  
his is a monotonic increasing word.  
accent is a nondecreasing word  
zone is a decreasing word.  
yucca is a nonincreasing word.  
-----------------------

$ egrep ^.{4}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort –n

$ shuf -ern 8000 {a..z}|xargs -L 4|sed 's/\ //g'|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -18

10 blls 212 1 eery 122  
 11 bbcz 122 1 ooze 120  
 11 hhfd 100 8 miff 001  
 16 agqq 221 19 bell 221  
 18 dccx 012 21 ally 212  
 22 aame 120 23 feed 010  
 24 ddbx 102 27 abba 210  
 25 ajjh 210 38 eddy 012  
 25 cabb 021 39 biff 201  
 28 amhh 201 47 ball 021  
 64 abcy 222 50 life 000  
 72 hfea 000 63 abet 222  
 197 bafn 022 174 able 220  
 205 ecbd 002 190 aged 200  
 206 abqj 220 202 fear 002  
 222 amja 200 248 babe 022  
 408 bazq 020 365 afar 202  
 417 aeaf 202 475 bake 020

$ paste <(shuf -ern 25000 {a..z}|xargs -L 6|sed 's/\ //g'|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -20) <(egrep ^.{6}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -20)

64 gazfec 02000 65 adagio 20222  
 70 acadpw 20222 66 health 00220  
 78 aglbgo 22022 67 backup 02220  
 78 awgfck 20002 69 abduce 22202  
 90 ihcxut 00200 101 amical 20002  
 101 dabnxd 02220 108 abrade 22022  
 115 atrauv 20022 108 cajole 02200  
 122 bamuih 02200 109 ballad 02102  
 124 abriet 22002 134 abased 20200  
 153 ebaltp 00220 146 abacus 20220  
 157 bahehv 02022 148 abject 22002  
 169 abnfwk 22020 148 alight 20022  
 171 akauob 20200 176 ablate 22020  
 189 asnlol 20020 181 afeard 20020  
 193 baqrbe 02202 183 featly 00202  
 199 gdayfp 00202 237 backer 02202  
 213 acadvq 20220 254 bakery 02022  
 231 caztov 02002 270 banger 02002  
 293 cawlnc 02020 317 agency 20202  
 304 abapcl 20202 346 balize 02020

02102 (ballad)

Compare its frequency (109 words out of 4321 six letter words) with the following based on a similar count ($ echo "4321 \* 6"|bc = 25926) of six letter random words:

$ shuf -ern 25926 {a..z}|xargs -L 6|sed 's/\ //g'|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|grep 02102

11 ihvviw 02102

$ egrep ^.{6}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|grep 02102

ballad 02102  
ballet 02102  
banner 02102  
barrel 02102  
barren 02102  
barrow 02102  
basset 02102  
batten 02102  
batter 02102  
caller 02102  
capper 02102  
carrot 02102  
dagger 02102  
dapper 02102  
fallen 02102  
farrow 02102  
fatten 02102  
fellah 02102  
fennel 02102  
ferret 02102  
fetter 02102  
gaffer 02102  
galley 02102  
gammer 02102  
garret 02102  
hammer 02102  
happen 02102  
harrow 02102  
hatter 02102  
jennet 02102  
kennel 02102  
killer 02102  
kipper 02102  
kisser 02102  
kitten 02102  
lammas 02102  
lappet 02102  
latter 02102  
lerret 02102  
lessen 02102  
lesser 02102  
lessor 02102  
letter 02102  
litter 02102  
mallet 02102  
mammal 02102  
manner 02102  
marrow 02102  
matter 02102  
miller 02102  
millet 02102  
mirror 02102  
mitten 02102  
mizzen 02102  
narrow 02102  
natter 02102  
nipper 02102  
pallet 02102  
parrot 02102  
passim 02102  
patten 02102  
patter 02102  
pellet 02102  
pepper 02102  
pillar 02102  
potter 02102  
powwow 02102  
rammer 02102  
rappel 02102  
rattan 02102  
reggae 02102  
rillet 02102  
rotten 02102  
rotter 02102  
sapper 02102  
seller 02102  
setter 02102  
simmer 02102  
sinner 02102  
sippet 02102  
sirrah 02102  
sitter 02102  
sorrel 02102  
sorrow 02102  
tanner 02102  
tassel 02102  
tatter 02102  
teller 02102  
tenner 02102  
tennis 02102  
terret 02102  
terror 02102  
tetter 02102  
tiller 02102  
tippet 02102  
titter 02102  
topper 02102  
totter 02102  
valley 02102  
vassal 02102  
vennel 02102  
vessel 02102  
wallet 02102  
warren 02102  
winner 02102  
yammer 02102  
yarrow 02102  
zaffer 02102  
zipper 02102

$ paste <(shuf -ern 41000 {a..z}|xargs -L 8|sed 's/\ //g'|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -20) <(egrep ^.{8}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -20)  
 59 cayeaeyh 0200220 59 apiarian 2002002  
 60 ageamwpu 2002202 61 abjectly 2200202  
 63 hdauctlc 0020200 63 babushka 0220020  
 64 ajcbsdsy 2002022 64 alarmist 2020022  
 65 cbfyfrsk 0220220 66 headland 0022020  
 67 abvpdzst 2200202 71 backdrop 0220202  
 69 afbudzfa 2020200 74 alacrity 2022022  
 75 ihaqpvuy 0020202 75 amenable 2020220  
 76 dbdcprnw 0202202 87 barbican 0202002  
 78 dcogogep 0202002 90 balister 0202202  
 81 acobrqsf 2202020 91 alkahest 2002022  
 82 ajfocfrp 2020220 93 acarpous 2020020  
 86 agcfxhvb 2022020 93 bargeman 0200202  
 87 baqnclkx 0200202 102 actively 2202022  
 91 canjpghr 0202022 123 acanthus 2022020  
 94 ajedfewl 2002020 132 ablation 2202020  
 94 aoevpozg 2020020 135 bakeshop 0202022  
 100 baetkvdp 0220202 139 alfresco 2020202  
 154 dcectqxe 0202020 145 alienage 2002020  
 166 ajgteico 2020202 227 balanced 0202020

$ egrep ^.{4}$ $w|awk 'BEGIN { C = "" ; for ( i = 0 ; ++i < 256 ; ) C = C sprintf ( "%c" , i ) };{for (i=1;i<NF;i++) {s=s"."(index(C,$(i+1))-index(C,$i))};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -20  
 2 hide .1.-5.1  
 2 john .5.-7.6  
 2 lean .-7.-4.13  
 2 link .-3.5.-3  
 2 lion .-3.6.-1  
 2 loaf .3.-14.5  
 2 loch .3.-12.5  
 2 meed .-8.0.-1  
 2 milt .-4.3.8  
 2 mold .2.-3.-8  
 2 molt .2.-3.8  
 2 opal .1.-15.11  
 2 open .1.-11.9  
 2 pail .-15.8.3  
 2 pelt .-11.7.8  
 2 proa .2.-3.-14  
 2 punk .5.-7.-3  
 2 spec .-3.-11.-2  
 3 abba .1.0.-1  
 3 lang .-11.13.-7

$ egrep ^.{4}$ $w|awk 'BEGIN { C = "" ; for ( i = 0 ; ++i < 256 ; ) C = C sprintf ( "%c" , i ) };{for (i=1;i<NF;i++) {s=s"."(index(C,$(i+1))-index(C,$i))};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|grep 1.0.-1  
abba deed noon.1.0.-1  
lang perk shun.-11.13.-7  
$ egrep ^.{3}$ $w|awk 'BEGIN { C = "" ; for ( i = 0 ; ++i < 256 ; ) C = C sprintf ( "%c" , i ) };{for (i=1;i<NF;i++) {s=s"."(index(C,$(i+1))-index(C,$i))};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|grep "\.3\.0"  
add bee ill loo.3.0  
  
$ egrep ^.{5}$ $w|awk 'BEGIN { C = "" ; for ( i = 0 ; ++i < 256 ; ) C = C sprintf ( "%c" , i ) };{for (i=1;i<NF;i++) {s=s"."(index(C,$(i+1))-index(C,$i))};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -5  
 1 zocle .-11.-12.9.-7  
 2 chain .5.-7.8.5  
 2 cheer .5.-3.0.13  
 2 opera .1.-11.13.-17  
 2 pecan .-11.-2.-2.13  
  
$ egrep ^.{5}$ $w|awk 'BEGIN { C = "" ; for ( i = 0 ; ++i < 256 ; ) C = C sprintf ( "%c" , i ) };{for (i=1;i<NF;i++) {s=s"."(index(C,$(i+1))-index(C,$i))};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|grep ".-11.-2.-2.13"  
Etc. for opera, cheer, chain  
pecan tiger .-11.-2.-2.13   
opera stive .1.-11.13.-17  
cheer jolly .5.-3.0.13  
chain ingot .5.-7.8.5

Bigger dictionary ($T)

$ egrep ^.{7}$ $T|awk 'BEGIN { C = "" ; for ( i = 0 ; ++i < 256 ; ) C = C sprintf ( "%c" , i ) };{for (i=1;i<NF;i++) {s=s"."(index(C,$(i+1))-index(C,$i))};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|tail -5

1 zymotic .-1.-12.2.5.-11.-6

1 zymurgy .-1.-12.8.-3.-11.18

1 zyzzyva .-1.1.0.-1.-3.-21

2 fortran .9.3.2.-2.-17.13 (FORTRAN)

2 primero sulphur.2.-9.4.-8.13.-3

steeds tuffet .1.-15.0.-1.15

paopao testes .-15.14.1.-15.14

inkier purply .5.-3.-2.-4.13

alohas grungy .11.3.-7.-7.18

anteed bouffe .13.6.-15.0.-1

pinot .-7.5.1.5 unsty .-7.5.1.5

mocha .2.-12.5.-7 suing .2.-12.5.-7

labor .-11.1.13.3 shivy .-11.1.13.3

ebola .-3.13.-3.-11 herod .-3.13.-3.-11

cobra .12.-13.16.-17 freud .12.-13.16.-17

banjo .-1.13.-4.5 ferns .-1.13.-4.5

$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS=""|sort -nk2|wc

86 172 516

$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS=""|head -30|xargs -L10

ace 3 act 3 ado 3 aft 3 ago 3 ail 3 aim 3 air 3 alp 3 amp 3  
ant 3 any 3 apt 3 art 3 beg 3 bel 3 ben 3 bet 3 bey 3 bin 3  
bis 3 bit 3 biz 3 bow 3 box 3 boy 3 buy 3 cop 3 cot 3 cow 3  
**Nondecreasing:**

**threes:**

$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i <= $(i+1)) c++ }; if (c>NF-2) print $0,NF }' FS=""|wc

102 204 612

(includes, for example, eel, inn and moo that are not strictly monotonic)

$ echo {a..z}| sed 's/[ ]/\*/g;s/z/z\*/'

a\*b\*c\*d\*e\*f\*g\*h\*i\*j\*k\*l\*m\*n\*o\*p\*q\*r\*s\*t\*u\*v\*w\*x\*y\*z\*  
$ grep ^`echo {a..z}| sed 's/[ ]/\*/g;s/z/z\*/'`$ $w|wc  
 310 310 1496

$ grep ^`echo {a..z}| sed 's/[ ]/\*/g;s/z/z\*/'`$ $w|xargs -L10|head -5  
a abbess abbey abbot abet abhor ably abort accent accept  
access accost ace act add adder adept adit ado adopt  
aegis affix afflux afoot aft agio aglow ago ah ail  
aim air airy all alloquy allot allow alloy ally almost  
alms alp am amp amps an annoy ant any apt

**Nonincreasing:**

$ grep ^`echo {z..a}| sed 's/[ ]/\*/g;s/a/a\*/'`$ $w|wc

196 196 900

Only 196 of these, as opposed to 310 nondecreasing

$ grep ^`echo {z..a}| sed 's/[ ]/\*/g;s/a/a\*/'`$ $w|xargs -L10|tail -5

unfed up upon urge urn us use used via vie  
void vomica we web wed wee weed wife wig wigged  
woe woke wold wolf womb won woo wood woof wool  
woon wrong x ye yea yob yoga yoke yolk yon  
yucca yule yuppie zone zoo zoom

So, for a random string of length three to be monotonic increasing, we must have all three chars distinct. Of the 26^3 = 17576 strings of length three, 26\* 25\* 24 of them have three distinct chars. So P(3 distinct) = 26\*25\*24/26^3 [≈](https://en.wikipedia.org/wiki/%E2%89%88) .888. Once three distinct chars are chosen, each of the six orderings (abc, acb, bac, bca, cab and cba) is equally likely, and only one is monotonic increasing. Hence the probability of getting three chars, at random, to be monotonic increasing is about .148 . The same would be true of the probability of having three chars being monotonic decreasing.

Given that there are 587 three letter words in $w \*, we’d expect (26\*25\*24/(6\*26^3))\*587 or about 86.83 to be monotonic increasing and the same number to be monotonic decreasing.

Sure enough, there are 86 increasing words:

$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0 }' FS=""|wc  
 86 86 344  
  
$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0 }' FS=""|tail -30|xargs -L15  
fry gin gnu got guy him hip his hit hop hot how hoy imp ivy  
jot joy lop lot low lox loy mop mow nor not now opt pry sty

But only 57 decreasing ones:

$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0 }' FS=""|wc  
57 57 228

$ egrep ^[a-z]{3}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0 }' FS=""|tail -30|xargs -L15  
sec she sib sic ski sob sod son spa tea ted the tic tie tod  
toe tom ton urn use via vie web wed wig woe won yea yob yon  
  
\* $ egrep ^[a-z]{3}$ $w|wc  
 587 587 2348  
examples:

$ egrep ^[a-z]{3}$ $w|tail -45|xargs -L15  
vim vow wad wag wan war was wat wax way web wed wee wem wen  
wet who why wig win wit woe won woo wop wot wry yak yam yap  
yaw yea yen yes yet yew yin yip yob yon you zap zip zit zoo

**Four letter words**

For four letters, the probability of four random letters being all different is

(26\*25\*24\*23/(26^4)) [≈](https://en.wikipedia.org/wiki/%E2%89%88).785 .

Once all four letters are different, the likelihood of being monotonically increasing would be 1/24 (given 4! permutations of the letters, with only one of those being as desired).

(26\*25\*24\*23/(26^4))/24[≈](https://en.wikipedia.org/wiki/%E2%89%88) .0327.

$ egrep ^[a-z]{4}$ $w|tail -45|xargs -L15   
word wore work worm worn wove wrap wren writ wynd yang yank yard yare yarn  
yarr yaup yawl yawn yawp yean year yell yelp yerk yeti yipe yoga yogi yoho  
yoke yolk yore your yule zany zarp zeal zebu zero zest zinc zone zoom zoot  
$ egrep ^[a-z]{4}$ $w|wc

1953 1953 9765

We would thus, expect about 1953 \* .0327[≈](https://en.wikipedia.org/wiki/%E2%89%88)63.89 of the four letter words to increase alphabetically.

Sure enough,

$ egrep ^[a-z]{4}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0 }' FS=""|wc  
 61 61 305  
  
$ egrep ^[a-z]{4}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0 }' FS=""|xargs -L16  
abet ably adit agio airy alms amps arty belt bent best bevy blot blow cent chin  
chip chit chop chow city clot cloy copy cost cosy crux deft defy demo dent deny  
dewy dint dirt dory doxy envy film fist flop flow flux fort foxy gilt gimp girt  
gist glow gory hilt hint hist hops host knot know lost most nosy

However, again, the reversals seem not to hold up their end of the probability distribution:

$ egrep ^[a-z]{4}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0 }' FS=""|wc  
 48 48 240  
  
$ egrep ^[a-z]{4}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0 }' FS=""|xargs -L16  
life mica olid pica pied plea poke pole pond rife role shed skid sled slid soda  
sofa soke sold sole some song spec sped spic tied toga told tomb tome tone tong  
trig trod upon urge used void wife woke wold wolf womb yoga yoke yolk yule zone

**Five letter words:**

$ egrep ^[a-z]{5}$ $w|wc  
 2892 2892 17352  
$ egrep ^[a-z]{5}$ $w|tail -36|xargs -L12   
worth would wound woven wrack wrath wreak wreck wrest wring wrist write  
wrong wrote wrung wryly xebec xenia xerox yacht yahoo yamen yearn yeast  
yield yodel yokel young yours youth yucca zambo zebra zilch zippo zocle

Increasing:

$ egrep ^[a-z]{5}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0 }' FS=""|xargs -L12

abhor abort adept adopt aegis aglow befit begin begot below bijou chimp

deist deity dirty empty filmy first forty ghost gipsy glory mopsy

]$ egrep ^[a-z]{5}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i < $(i+1)) c++ }; if (c>NF-2) print $0 }' FS=""|wc

23 23 138

Decreasing:

$ egrep ^[a-z]{5}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0 }' FS=""|wc

8 8 48

$ egrep ^[a-z]{5}$ $w|awk '{c=0; for (i=1; i<=NF; ++i) { if ($i > $(i+1)) c++ }; if (c>NF-1) print $0 }' FS=""|xargs -L12

polka solid sonic spoke theca tonic unfed wrong

Expectation:

About 2/3 of 5 letter sequences would have all five letters different:

((26\*25\*24\*23\*22/(26^5))) [≈](https://en.wikipedia.org/wiki/%E2%89%88) .6644.

But those 5 letters must all be in the proper order (which happens with probability only 1/5! or 1/120 )

((26\*25\*24\*23\*22/(26^5))/120) [≈](https://en.wikipedia.org/wiki/%E2%89%88) 0.005536

With 2892 five letter words, then we’d expect

((26\*25\*24\*23\*22/(26^5))/120)\* 2892 [≈](https://en.wikipedia.org/wiki/%E2%89%88) 16.011 for both increasing and decreasing.

Are variations as wide as 23 (increasing) and 8 (decreasing) within the realm of randomness?

Here are some random trials. The script generates 14460 chars in 2892 groups of five letter words and then sorts the words based on their internal rhythms (see more on this topic later). We restrict the output to the strictly increasing sequences (2222) or the scrictly decreasing ones (0000). A few trials are run just to give an idea

]$ shuf -ern 14460 {a..z}|xargs -L 5|sed 's/\ //g'|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|egrep "([02])\1\1\1"

10 aboqy 2222  
 15 rkiha 0000  
 12 nhgcb 0000  
 18 adkpq 2222  
 12 aisuw 2222  
 18 tngfe 0000  
 11 acfst 2222  
 23 igfba 0000  
 13 aglvx 2222  
 13 jhfea 0000  
 15 nmjhf 0000  
 18 acimz 2222  
 14 mihfa 0000  
 16 adflr 2222  
 8 adhtz 2222  
 18 roidc 0000

2892 \* 5 = 14460

Sure enough, variations as wide as observed among real words are seen as entirely possible within the laws of chance.

**Six**

((26\*25\*24\*23\*22\*21/(26^6))) [≈](https://en.wikipedia.org/wiki/%E2%89%88) 0.5366

((26\*25\*24\*23\*22\*21/(26^6)))/720 [≈](https://en.wikipedia.org/wiki/%E2%89%88) 0.00074528404

$ egrep ^[a-z]{6}$ $w|wc

4278 4278 29946

4278\*((26\*25\*24\*23\*22\*21/(26^6)))/720 [≈](https://en.wikipedia.org/wiki/%E2%89%88) 3.188 = expected number of monotonic (up or down) sequences for six letter strings.

$ expr 4278 "\*" 6

25668

$ shuf -ern 25668 {a..z}|xargs -L 6|sed 's/\ //g'|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|egrep "([012])\1\1\1\1"

4 aejkls 22222  
 5 utokdc 00000  
 2 abdhrs 22222  
 3 ysonga 00000  
 3 abltuv 22222  
 3 wtoldc 00000  
 1 eimpqv 22222  
 2 vupkga 00000  
 0 00000  
 2 cefjmy 22222  
 4 omjfea 00000  
 6 aekqtx 22222  
 3 ahmotv 22222  
 3 toieca 00000  
 4 pmhgfd 00000  
 5 adfhln 22222

$ egrep ^[a-z]{6}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|egrep "([012])\1\1\1\1"

2 sponge 00000

5 almost 22222

**Seven**

((26\*25\*24\*23\*22\*21\*20/(26^7)))[≈](https://en.wikipedia.org/wiki/%E2%89%88) 0.4128  
((26\*25\*24\*23\*22\*21\*20/(26^7)))/5040 [≈](https://en.wikipedia.org/wiki/%E2%89%88) 0.0000819  
$ egrep ^[a-z]{7}$ $w|wc  
 4854 4854 38832  
4854 \* ((26\*25\*24\*23\*22\*21\*20/(26^7)))/5040 [≈](https://en.wikipedia.org/wiki/%E2%89%88) 0.3975= expected number of monotonic (up or down) sequences for seven letter strings.

$ expr 4278 "\*" 6

25668

$ egrep ^[a-z]{7}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|egrep "([012])\1\1\1\1"  
 1 dyspnea 200000  
 2 obloquy 022222  
 2 polecat 000002  
$ egrep ^[a-z]{7}$ $w|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|egrep "([012])\1\1\1\1"  
dyspnea 200000  
obloquy 022222  
polecat 000002  
sponger 000002  
thirsty 022222  
Demonstrates that there are no strictly monotonic sequences of length 7 in $w. In fact there are none of length seven or higher.

$ wc $T $w  
 406712 406712 4158156 /home/ddailey/public\_html/moby/mthes/TwoOrMore  
 35916 35916 332173 /home/ddailey/public\_html/words  
  
  
In the much larger dictionary ($T), there are a couple:

$ egrep ^[a-z]{7}$ $T|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|sort -k2|uniq -cf1|sort -n|egrep "([012])\1\1\1\1"  
 2 deglory 222222  
 2 sponged 000000  
 16 bailors 022222  
 19 lifeday 000002  
 22 avonlea 200000  
 25 abortus 222220  
$ egrep ^[a-z]{7}$ $T|awk '{for (i=1;i<NF;i++) {if ($i>$(i+1))s=s""0;else if ($i<$(i+1))s=s""2;else s=s""1};{print s" "$0;s="";}} ' FS=""|awk '{print $2,$1}'|egrep "([012])\1\1\1\1\1"  
deglory 222222  
egilops 222222  
sponged 000000  
wronged 000000  
and

[truncated?]

**Counting chars in dict:**

$ grep -o . $w|wc -l  
296257  
$ wc $w  
 35916 35916 332173 /home/ddailey/public\_html/words  
$ expr 296257 + 35916  
332173

Vowels:  
  
$ grep -o "[aeiou]" $w|wc -l  
114419  
$ grep -io "[aeiou]" $w|wc -l  
114444  
$ grep -o "[AEIOU]" $w|wc -l  
25  
25 + 114419 = 114444  
  
Consonants  
  
$ grep -o "[bcdfghjklmnpqrstvwxyz]" $w|wc -l  
180896  
$ grep -oi "[bcdfghjklmnpqrstvwxyz]" $w|wc -l  
180944  
$ grep -o "[BCDFGHJKLMNPQRSTVWXYZ]" $w|wc -l  
48  
$ expr 48 + 180896  
180944  
  
Together:  
$ grep -io "[aeiou]" $w|wc -l  
114444  
$ grep -oi "[bcdfghjklmnpqrstvwxyz]" $w|wc -l  
180944  
  
$ grep -o . $w|wc -l  
296257  
$ expr 114444 + 180944  
295388

Nonalphabetic characters:  
$ grep -oi "[^a-z]" $w|wc  
 869 869 1738

$ grep -oi "[^a-z]" $w|sort|uniq -c  
 746 -  
 30 ;  
 1 .  
 62 '  
 30 &  
$ expr 746 + 30 + 1 + 62 + 30  
869

$ expr 869 + 295388  
296257

This shows a partition of the 296257 characters of $w = /home/ddailey/public\_html/words into:

Vowels: 114444  
Consonants: 180944  
And other: 869

$ wc /home/ddailey/public\_html/moby/mthes/SixOrMore

66023 66023 595432 /home/ddailey/public\_html/moby/mthes/SixOrMore

$ echo {A..Z} {a..z}|sed s/[aeiouAEIOU\ ]//g

BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz

$ paste <(head SixOrMore) <( head SixOrMore |sed 's/[aeiouAEIOU]/A/g;s/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g')  
a V  
a- V-  
A V  
aa VV  
aah VVC  
aahs VVCC  
aardvark VVCCCVCC  
aardwolf VVCCCVCC  
aas VVC  
ab VC

$ cat SixOrMore|sed -n '/^....$/s/[aeiouAEIOU]/A/gp'|sed 's/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g'|sort|uniq -c|sort -nr  
 1227 CVCC  
 662 CVCV  
 468 CVVC  
 410 CCVC  
 150 VCVC  
 68 VCCV  
 59 VCCC  
 49 CCVV  
 38 VVCC  
 32 CCCV  
 18 VCVV  
 10 VVCV  
 9 CVVV  
 3 V'VC  
 2 CVC-  
 2 CCV-  
 1 VVVC  
 1 'CVC

$ cat SixOrMore|sed -n '/^.....$/s/[aeiouAEIOU]/A/gp'|sed 's/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g'|sort|uniq -c|sort -nr

1340 CVCVC  
 908 CVCCC  
 721 CCVCC  
 507 CVCCV  
 490 CVVCC  
 303 CCVCV  
 297 CCVVC  
 247 VCCVC  
 133 VCVCC  
 123 CVVCV  
 118 VCVCV  
 107 CVCVV  
 78 CCCVC  
 69 VCVVC  
 45 VVCVC  
 25 VCCVV  
 24 CVVVC  
 21 VCCCV  
 20 VCCCC  
 17 CCCCV  
 12 VVCCC  
 9 CCCVV  
 7 VVCCV  
 5 CV-CV  
 4 CVC'C  
 3 VVCVV  
 3 CVVVV  
 2 CV'VC  
 2 CVCC-  
 1 VVC'C  
 1 VCVVV  
 1 VCV'C  
 1 VCC'C  
 1 CV-VC  
 1 CVïCV  
 1 CVCV-  
 1 CV'CV  
 1 CV-CC  
 1 C-CVC  
 1 'CCVC

cat SixOrMore|sed -n '/^......$/s/[aeiouAEIOU]/A/gp'|sed 's/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g'|sort|uniq -c|sort -nr|head -50

2308 CVCCVC  
 905 CVCVCC  
 620 CCVCVC  
 501 CCVCCC  
 497 CVVCVC  
 492 CVCVCV  
 380 CVCVVC  
 328 VCCVCC  
 257 CCVVCC  
 239 CVCCCV  
 193 VCCVCV  
 180 VCVCVC  
 178 CVVCCC  
 157 CVCCVV  
 151 CVCCCC  
 128 CCVCCV  
 125 VCCVVC  
 111 CCCVCC  
 104 VCCCVC  
 103 CVVCCV  
 64 CCVVCV  
 59 CCCVCV  
 57 CCCCVC  
 49 VVCCVC  
 45 VCVCCC  
 41 CVVCVV  
 40 VCVVCC  
 37 VCVCCV  
 35 CCCVVC  
 31 VVCVCC  
 28 CCVCVV  
 21 VCVVCV  
 18 VCVCVV  
 17 VVCVCV  
 14 CVVVCC  
 9 VCCCVV  
 8 CCCCCV  
 7 VVCVVC  
 7 VVCCCC  
 7 VCCCCV  
 5 CVCVVV  
 5 CCCCVV  
 4 CCVVVC  
 3 CVVVCV  
 3 CVCC'C  
 2 VVCCVV  
 2 VVCCCV  
 2 VCCCCC  
 2 CVVVVC  
 2 CV-VCC

Seven letters:

$ cat SixOrMore|sed -n '/^.......$/s/[aeiouAEIOU]/A/gp'|sed 's/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g'|sort|uniq -c|sort -nr|head -50

1824 CVCCVCC  
 928 CCVCCVC  
 821 CVCVCVC  
 694 CVCCCVC  
 623 CVCCVCV  
 546 CVCCVVC  
 394 CVVCVCC  
 361 CVVCCVC  
 360 CCVCVCC  
 333 VCCVCVC  
 263 CCVVCVC  
 231 CVCVCCV  
 195 CVCVCCC  
 167 CCVCVCV  
 166 CVCVVCC  
 147 VCVCVCC  
 137 CCVCVVC  
 136 VCCCVCC  
 125 VCVCVCV  
 111 VCVCCVC  
 111 VCCVCCC  
 110 CCVCCCV  
 89 CVCVVCV  
 88 VCCVVCC  
 87 CCCVCVC  
 85 VCCCVCV  
 85 CVVCVCV  
 82 VCCVCCV  
 82 CCVCCCC  
 78 CVCVCVV  
 72 CCVVCCC  
 65 CVVCVVC  
 62 VCVCVVC  
 60 VCCCVVC  
 47 VVCCVCC  
 44 CCCVCCC  
 40 CVCVVVC  
 36 CCCCVCC  
 35 CVVCCCC  
 34 VCVVCVC  
 30 VCCVVCV  
 28 CCVCCVV  
 26 VCCCCVC  
 26 CCVVCCV  
 25 VVCCCVC  
 25 CCCVVCC  
 24 CCCCCVC  
 22 VVCVCVC  
 22 VCCVCVV  
 22 CCCCVVC

Eight letters

$ cat SixOrMore|sed -n '/^........$/s/[aeiouAEIOU]/A/gp'|sed 's/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g'|sort|uniq -c|sort -nr|head -50

927 CVCCVCVC  
 835 CVCVCVCC  
 742 CCVCCVCC  
 623 CVCCCVCC  
 462 CVCVCCVC  
 422 CVCVCVCV  
 370 CVCVCVVC  
 349 CVVCCVCC  
 332 VCCVCCVC  
 328 CVCCVCCC  
 268 VCCVCVCC  
 261 CCVCCCVC  
 239 CCVCVCVC  
 227 CCVVCVCC  
 192 CVCCVVCC  
 192 CVCCVCCV  
 191 CVCCCVCV  
 182 CVCCCVVC  
 141 VCCVCVCV  
 135 VCCCVCVC  
 132 CVCVVCVC  
 123 VCVCVCVC  
 121 VCVCCVCC  
 120 CVCCCCVC  
 116 CVVCVCVC  
 115 CCVCCVVC  
 111 CCVCCVCV  
 105 VCCVCVVC  
 92 CVVCCCVC  
 90 CCVVCCVC  
 85 CVVCCVCV  
 84 CCCVCCVC  
 77 VCCVVCVC  
 75 CVVCCVVC  
 74 VCVCCVCV  
 67 CVVCVCCC  
 65 CVCCVVCV  
 61 VCVCCVVC  
 60 CCVCVCCC  
 57 CVCVCCCC  
 56 CCVCVCCV  
 52 CVCVVCCV  
 52 CVCCVCVV  
 47 CVCVVCCC  
 43 CVVCVCCV  
 42 CCCVCVCC  
 41 VCVCCCVC  
 38 CCVCVVCC  
 38 CCCCVCVC  
 37 CCVVCVCV

Spanish

$echo $s  
es.txt  
data$pwd  
/home/SRUNET/david.dailey/data

Most frequent characters

$cat $s|sed 's/\ .\*//;s/./&\n/g'|awk '!/^$/'|sort|uniq -c|sort -nr|head -50

537718 a  
 454007 e  
 353327 r  
 342698 o  
 336226 i  
 313406 s  
 295433 n  
 224557 t  
 215427 l  
 189358 c  
 165198 d  
 148208 m  
 135614 u  
 101579 p  
 82573 b  
 79228 g  
 69799 h  
 52981 v  
 46424 f  
 37876 k  
 30150 y  
 29973 á  
 29960 z  
 25592 í  
 25280 j  
 24380 é  
 17352 ó  
 14592 w  
 14034 q  
 10128 x  
 4989 ñ  
 4261 ú  
 2045 ò  
 1754 à  
 1732 ô  
 1588 â  
 1441 ï  
 1170 è  
 885 ü  
 721 ì  
 701 ê  
 591 ｿ  
 466 ã  
 438 ö  
 401 ż  
 396 ą  
 358 ç  
 317 î  
 309 ä  
 247 û

$grep ｿ $s  
ｿpor 16  
ｿest疽 16  
ｿte 12  
ｿno 12  
ｿpuedo 8  
ｿde 8  
ｿeres 7  
ｿes 6  
ｿqui駭 6

A Google search for ‘ｿest疽’ reveals about 5000 hits, including

<https://commons.wikimedia.org/wiki/TimedText:The_Million_Ryo_Pot_(1935).webm.ja.srt>

# Entitled “Japanese subtitles for clip: File:The Million Ryo Pot (1935).webm” , the page has 1219 entries, many of which appear to be Spanish with frequent transcription errors: e.g.

725

00:52:24,546 --> 00:52:27,276

Es la segunda casa desde la esquina,

delante de un pozo. No tiene p駻dida.

# $cat $s|sed 's/\ .\*//;s/./&\n/g'|awk '!/^$/'|sort|uniq -c|sort -nr|head -50|awk '{print $2}'|tr '\n' ' ' a e r o i s n t l c d m u p b g h v f k y á z í j é ó w q x ñ ú ò à ô â ï è ü ì ê ｿ ã ö ż ą ç î ä û a$v=[aeoiuáíéóúòàôâïèüìêãöąîäû] data$c=[rsntlcdmpbghvfkyzjwqxñżç] Spanish 4:

# $awk '{print $1}' $s|sed -n '/^.\{4\}$/s/[aeoiuáíéóúòàôâïèüìêãöąîäû]/V/gp'|sed 's/[rsntlcdmpbghvfkyzjwqxñżç]/C/g'|sort|uniq -c|sort -nr|head -24

# 6082 CVCV 3453 CVCC 2066 CVVC 1865 CCVC 1520 VCVC 1251 VCCV 568 CCCV 562 CVVV 506 VCVV 498 CCVV 467 VVCV 441 VCCC 165 VVCC 124 VVVC 62 VVVV 14 ｿCVC 13 CVCž 9 CVńV 8 ĺźCV 8 CVëC 6 CVCแ 5 CVýV 5 CVCù 5 CV

# English 4

# $cat $e|sed 's/\ .\*//;s/./&\n/g'|awk '!/^$/'|sort|uniq -c|sort -nr|head -50|awk '{print $2}'|tr '\n' ' ' $cat $e|sed 's/\ .\*//;s/./&\n/g'|awk '!/^$/'|sort|uniq -c|sort -nr|head -50|awk '{print $2}'|tr '\n' ' ' e a i r o n s t l u c h d m g p b k y f v w z j x q é ÿ í á ï ä ó è ö ñ î þ ã а ü о е и ç å à ý ê т data$ev="e a i o u é ÿ í á ï ä ó è ö î ã а ü о е и å à ê" data$ec="r n s t l c h d m g p b k y f v w z j x q ñ þ ç т" data$echo $ec|sed 's/\ //g' rnstlchdmgpbkyfvwzjxqñþçт data$echo $ev|sed 's/\ //g' eaiouéÿíáïäóèöîãаüоеиåàê

# $awk '{print $1}' $e|sed -n '/^.\{4\}$/s/[eaiouéÿíáïäóèöîãаüоеиåàê]/V/gp'|sed 's/[rnstlchdmgpbkyfvwzjxqñþçт]/C/g'|sort|uniq -c|sort -nr|head -2

# 5235 CVCV 5079 CVCC 2704 CCVC 2500 CVVC 1651 VCVC 1269 VCCV 884 VCCC 881 CCCV 568 CCVV 514 CVVV 437 VVCC 420 VVCV 413 VCVV 172 VVVC 51 VVVV 18 CôCV 14 CVCô 13 ηVCC 12 CVšV 12 CVCò 11 CøCV 10 CâCV 9 CVCú 8 žVCV

# Note that for four letter words, in both Spanish and English, CVCV is the top-occuring pattern, while CVCC is second. Note also that when I used the top fifty characters in English ‘ô’ and ‘ú’ clearly vowels didn’t appear in the top fifty. The above script could clearly be refined, but it is interesting to note that the pattern CôCV is slightly more frequent than VCCC or CCCC in this particular vocabulary of the language. (some of the more frequent occurances: $grep "^.ô..\ " $e côte 17 (as in Côte d’Azur), môle 14, côté 13, dôme 10, môme 6, cômo 5, côme 5, rôti 4 (as in poulet rôti - wrapped in bacon, with purée and fennel (https://www.tripadvisor.co.uk/LocationPhotoDirectLink-g186338-d1388950-i94968576-Cote\_Brasserie\_Covent\_Garden-London\_England.html) ), also in familiar appearance: lancôme 3,)

# French and German (just for fun):

# $wc $g $f 317388 634776 4573651 de.txt 305763 611526 3833939 fr.txt 623151 1246302 8407590 total

# $cat $f $g|sed 's/\ .\*//;s/./&\n/g'|awk '!/^$/'|sort|uniq -c|sort -nr|head -60|awk '{print $2}'|tr '\n' ' ' e r n i a s t l o u h c g m d p b f k v é z w y ä ü j q x è ö ê ß ï í â î ç ô á ž û à ó ì č š å ã ë ο ñ ú œ ę þ ù æ ÿ õ

# $echo $fgv eiasouéyäüèöêïíâîôáûàóìåãëοúœęùæÿõ $echo $fgc|sed 's/\ //g' rnstlhcgmdpbfkvzwyjqxßçžčšñþ

# $awk '{print $1}' $f|sed -n '/^.\{4\}$/s/[eiasouéyäüèöêïíâîôáûàóìåãëοúœęùæÿõ]/V/gp'|sed 's/[rnstlhcgmdpbfkvzwyjqxßçžčšñþ]/C/g'|sort|uniq -c|sort -nr|head -24

# 4184 CVCV 1720 CVCC 1597 CVVC 1171 CVVV 945 CCVC 903 VCVC 864 VVCV 838 VCCV 680 VCVV 553 CCVV 381 VVVC 304 VVVV 283 CCCV 272 VVCC 165 VCCC 5 CVCò 5 CòCV 5 CˆCV 4 CVďC 3 VCCò 3 CVC嶪 3 CVCŕ 3 CVCø 3 CøCV

# German $awk '{print $1}' $g|sed -n '/^.\{4\}$/s/[eiasouéyäüèöêïíâîôáûàóìåãëοúœęùæÿõ]/V/gp'|sed 's/[rnstlhcgmdpbfkvzwyjqxßçžčšñþ]/C/g'|sort|uniq -c|sort -nr|head -24

# 2376 CVCV 1793 CVCC 1151 CVVC 729 CCVC 648 CVVV 635 VCVC 498 VCCV 474 VVCV 311 VCVV 291 VVCC 268 CCVV 251 VVVC 176 VCCC 169 VVVV 144 CCCV 4 κVCC 3 ηVCC 2 μVCC 2 ηVVC 2 ηVCV 2 εVCV 2 αVCV 2 αCVV 1 νVVC

# Conclusion: It is interesting to note that for these four languages, the most prevalent forms of Consonant-Vowel rhythms for four letter words are, first: CVCV and second: CVCC).

# English 5

# $awk '{print $1}' $e|sed -n '/^.\{5\}$/s/[eaiouéÿíáïäóèöîãаüоеиåàê]/V/gp'|sed 's/[rnstlchdmgpbkyfvwzjxqñþçт]/C/g'|sort|uniq -c|sort -nr|head -32

# 10562 CVCVC 8473 CVCCV 4573 CVCCC 3367 CCVCC 2736 CCVCV 2623 CVVCV 2550 VCCVC 2533 CVCVV 2403 CVVCC 1888 VCVCV 1419 CCVVC 1319 CCCVC 1075 VCVCC 583 CVVVC 568 VCVVC 534 VVCVC 512 CCCCV 507 VCCVV 490 VCCCV 337 VCCCC 243 VVCCC 216 VVCCV 214 CCCVV 170 CCVVV 126 VVVCC 109 CVVVV 105 VVCVV 74 VVVCV 67 VCVVV 37 VVVVC 25 VVVVV 17 CVυCC

# Spanish 5

# $awk '{print $1}' $s|sed -n '/^.\{5\}$/s/[aeoiuáíéóúòàôâïèüìêãöąîäû]/V/gp'|sed 's/[rsntlcdmpbghvfkyzjwqxñżç]/C/g'|sort|uniq -c|sort -nr|head -32

# 10524 CVCVC 8301 CVCCV 3180 CVCVV 2962 CVVCV 2911 VCVCV 2743 CCVCV 2540 CVCCC 2080 VCCVC 1702 CCVCC 1247 CVVCC 904 CCVVC 657 CCCVC 578 CVVVC 577 VCCVV 555 VCVVC 451 VCVCC 437 VVCVC 381 VCCCV 339 VVCCV 272 CCCCV 248 CVVVV 149 VVVCV 136 VVCVV 133 CCVVV 132 CCCVV 117 VCCCC 106 VCVVV 77 VVCCC 48 VVVVC 37 VVVVV 29 VVVCC 13 ｿCVCV

# Note that for five letter words, in both Spanish and English, CVCVC is the top-occuring pattern, while CVCCV is second.

# Spanish 6

# $awk '{print $1}' $s|sed -n '/^.\{6\}$/s/[aeoiuáíéóúòàôâïèüìêãöąîäû]/V/gp'|sed 's/[rsntlcdmpbghvfkyzjwqxñżç]/C/g'|sort|uniq -c|sort -nr|head -32

# 13257 CVCCVC 13213 CVCVCV 3243 CVCVVC 3210 CCVCVC 3127 CVVCVC 2832 CVCCVV 2777 VCCVCV 2769 CVCVCC 2394 VCVCVC 2038 CCVCCV 1994 CVCCCV 1699 CVVCCV 1671 VCVCCV 910 CCVCCC 780 CCVVCV 767 CCVCVV 750 VCVCVV 730 VCCVCC 676 CVVCVV 647 VCCCVC 605 CVCCCC 596 VCCVVC 579 VCVVCV 509 CCVVCC 468 CVCVVV 432 CCCVCV 414 CVVVCV 396 VVCVCV 375 CCCVCC 343 CVVCCC 338 CCCCVC 270 VVCCVC

# English 6

# $awk '{print $1}' $e|sed -n '/^.\{6\}$/s/[eaiouéÿíáïäóèöîãаüоеиåàê]/V/gp'|sed 's/[rnstlchdmgpbkyfvwzjxqñþçт]/C/g'|sort|uniq -c|sort -nr|head -32

# 15464 CVCCVC 8722 CVCVCV 4935 CVCVCC 3706 CCVCVC 3489 CVVCVC 2623 CVCCVV 2578 CVCCCV 2559 CVCVVC 2333 CCVCCV 1980 CCVCCC 1841 VCCVCC 1696 VCCVCV 1602 CVCCCC 1560 CVVCCV 1454 VCVCVC 1107 CCVVCC 981 CCCVCC 968 VCCCVC 943 CVVCCC 915 VCVCCV 843 CCCCVC 777 CVVCVV 737 CCVVCV 698 VCCVVC 653 CCCVCV 635 CCVCVV 420 VVCCVC 396 CCCVVC 366 VCVCVV 319 VCVCCC 317 VCVVCV 280 VVCVCC

# Note that for six letter words, in both Spanish and English, CVCCVC is the top-occuring pattern, while CVCVCV is second. However, note some disagreement in lower ranked patterns:

# English (4935 CVCVCC)3 > (2559 CVCVVC)8

# Spanish (2769 CVCVCC)8 < (3243 CVCVVC)3

# English 7

# $awk '{print $1}' $e|sed -n '/^.\{7\}$/s/[eaiouéÿíáïäóèöîãаüоеиåàê]/V/gp'|sed 's/[rnstlchdmgpbkyfvwzjxqñþçт]/C/g'|sort|uniq -c|sort -nr|head -32

# 8710 CVCCVCC 6206 CVCCVCV 6140 CVCVCVC 5013 CVCCCVC 4789 CCVCCVC 4524 CVCVCCV 3145 CVCCVVC 2297 CVVCCVC 1789 CVVCVCC 1742 CCVCVCV 1707 CCVCVCC 1694 VCCVCVC 1463 CVCVCVV 1288 CCVVCVC 1242 CVCVVCV 1152 CVCVCCC 1091 VCVCVCV 1016 CVVCVCV 869 VCVCCVC 839 VCCVCCV 824 CVCVVCC 753 CCVCVVC 737 VCCCVCC 716 CCVCCCV 676 CVVCVVC 669 CCCVCVC 589 CVCCCCC 586 CCVCCVV 578 CCVCCCC 576 VCVCVCC 557 CVCCCVV 556 CCCCVCC

# Spanish 7

# $awk '{print $1}' $s|sed -n '/^.\{7\}$/s/[aeoiuáíéóúòàôâïèüìêãöąîäû]/V/gp'|sed 's/[rsntlcdmpbghvfkyzjwqxñżç]/C/g'|sort|uniq -c|sort -nr|head -32

# 10517 CVCVCVC 9802 CVCCVCV 7033 CVCVCCV 4554 CVCCVCC 3273 CVCCCVC 3047 CCVCCVC 2997 VCVCVCV 2932 CVCCVVC 2729 CCVCVCV 2579 VCCVCVC 2540 CVCVCVV 2328 CVCVVCV 1860 CVVCVCV 1805 CVVCCVC 1755 VCCVCCV 1225 VCVCCVC 845 CCVVCVC 763 VCCVCVV 762 CCVCVCC 741 CCVCVVC 735 CVVCVCC 713 VCVCVVC 642 VCCCVCV 608 VCCVVCV 564 CVVCVVC 512 CVCVCCC 502 CCVCCVV 469 CCVCCCV 447 VCVVCVC 442 CVCCCVV 439 CVCVVVC 380 CCCVCVC

# Let’s also look at French and German:

# French 7

# $awk '{print $1}' $f|sed -n '/^.\{7\}$/s/[eiasouéyäüèöêïíâîôáûàóìåãëοúœęùæÿõ]/V/gp'|sed 's/[rnstlhcgmdpbfkvzwyjqxßçžčšñþ]/C/g'|sort|uniq -c|sort -nr|head -24

# 3774 CVCCVCV 2838 CVCVCCV 2692 CVCVCVV 2300 CVCVCVC 1876 CVCCVCC 1874 CVCVVCV 1486 CVCCVVC 1276 CVVCVCV 1136 CVCCVVV 1080 CVCCCVC 1017 VCVCVCV 1013 CCVCVCV 931 VCCVCVV 915 CVVCCVC 898 CCVCCVC 893 VCCVCCV 804 CVVCCVV 754 CVCVVVV 680 VCVCCVC 667 CCVCCVV 663 CVCCCVV 642 VCCVCVC 630 CVVCVVC 611 CVVCVCC

# German 7

# $awk '{print $1}' $g|sed -n '/^.\{7\}$/s/[eiasouéyäüèöêïíâîôáûàóìåãëοúœęùæÿõ]/V/gp'|sed 's/[rnstlhcgmdpbfkvzwyjqxßçžčšñþ]/C/g'|sort|uniq -c|sort -nr|head -24

# 2214 CVCCVCV 2166 CVCCVCC 1577 CVCVCVC 1304 CVCCCVC 1234 CVCVCCV 967 CCVCCVC 911 CVCCVVC 846 CVVCCVC 762 CVCVCVV 686 VCCVCVC 665 VCVCCVC 650 CVVCVCV 642 CVCVVCV 620 CVVCVCC 565 CVCCVVV 564 VCCVCCV 461 CVCVCCC 453 CCVVCVC 393 VCVCVCV 393 CVCVVCC 390 VCCCVCC 376 CVVVCVC 367 CCVCVCV 354 CVCCCVV

# Note that English ( 8710 CVCCVCC)1 > (6140 CVCVCVC)3

# While Spanish (4554 CVCCVCC)4 < (10517 CVCVCVC)1

# In French (1876 CVCCVCC) 5 < (2300 CVCVCVC)4

# And in German (2166 CVCCVCC)2 > (1577 CVCVCVC) 3

# Examples:

|  |  |  |
| --- | --- | --- |
| CVCrhythm | English | Spanish |
| **CVCCVCC** | forward/selling | raymond/bistecs |
| **CVCCVCV** | destiny/lottery | soldado/cerrado |
| CVCVCVC | related/titanic | sigamos/pedimos |
| **CVCCCVC** | **matches/seltzer** | **mostrar/manchas** |
| **CCVCCVC** | **stalled/bracket** | **francos/prestar** |
| **CVCVCCV** | **bizarre/syringe** | **podréis/cambios** |
| **CVCCVVC** | **passion/penguin** | **viernes/sientas** |
| **CVVCCVC** | **neither/measles** | **cierren/cuernos** |
| **VCVCVCV** | **ability/episode** | **apetece/editado** |
| **CVCVCVV** | **someday/referee** | refería/delicia |
| **CVCVVCV** | **genuine/release** | líquido/valiosa |
| **CVVCVCV** | **sausage/seizure** | **realeza/quemado** |
| **CVCCVVV** | **kumbaya/hawkeye** | desmayó/turquía |
| **CCVCVCV** | **closely/precise** | llamaba/trasera |
| **VCCVCVV** | amnesia/antique | odiaría/acuario |

# Seven letter sequences: comparisons of consonant-vowel rhythms across English, Spanish, French and German:

# Relative frequency for most popular CVC sequences relative to the total number sampled.

# The above table involved first choosing the eight most frequently occurring sequences in English, and then “bootstrapping” outward so that each language’s highest frequency entries were included.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CVCCVCC** | **CVCCVCV** | CVCVCVC | **CVCCCVC** | **CCVCCVC** | **CVCVCCV** | **CVCCVVC** | **CVVCCVC** | **VCVCVCV** | **CVCVCVV** | **CVCVVCV** | **CVVCVCV** | **CVCCVVV** | **CCVCVCV** | **VCCVCVV** |
| English | 8710 | 6206 | 6140 | 5013 | 4789 | 4524 | 3145 | 2297 | 1454 | **1463** | 1241 | 1016 | 222 | **1742** | 366 |
| Spanish | **4554** | **9802** | 10517 | **3273** | **3047** | **7033** | **2932** | **1805** | **2997** | **2540** | **2328** | **1860** | **310** | **2729** | 734 |
| French | **1876** | 3774 | 2309 | **1080** | **1486** | **2838** | **1486** | **915** | **1017** | **2692** | 1874 | **1276** | **1136** | **1013** | **931** |
| German | **2166** | **2214** | **1577** | **1304** | **967** | **1234** | **911** | **846** | **393** | **762** | **642** | **650** | **565** | **367** | **704** |

# Specifically, if as we see above,In order to “get to” the eight highest sequences for English (CVVCCVC at 2297 in English but only 1805 in Spanish) the following Spanish sequences were higher in frequency than the Spanish value of this pattern: 1805. Namely, the sequences (VCVCVCV:2997, VCVCVCV:2540, CVCVCVV:2328, CVCVVCV: 1860) all had to be considered before inclusion of CVVCCVC could be entertained. This method was extended until all four languages had represented in the table, their top eight values. This required the addition of seven more columns as can be seen.

# Spanish 9

# $awk '{print $1}' $s|sed -n '/^.\{9\}$/s/[aeoiuáíéóúòàôâïèüìêãöąîäû]/V/gp'|sed 's/[rsntlcdmpbghvfkyzjwqxñżç]/C/g'|sort|uniq -c|sort -nr|head -32

# 6006 CVCVCVCVC 5452 CVCCVCVCV 4164 CVCVCCVCV 3651 CVCVCVCCV 3380 CVCCVCCVC 2702 VCCVCVCVC 1961 VCCVCCVCV 1868 VCCVCVCCV 1808 CVCCVCVVC 1420 CCVCVCVCV 1355 VCVCCVCVC 1297 CVCCVVCVC 1218 CVCCCVCVC 1136 VCVCVCVCV 1039 CVCVCVCVV 1002 CVCVCCVVC 989 CVCVCVVCV 946 CVCCVVCCV 939 CCVCCVCVC 933 VCVCCVCCV 741 CCVCVCCVC 740 VCCCVCVCV 734 CVCCCVCCV 705 CVCCVCVCC 644 CVVCVCVCV 627 VCVCVCCVC 612 CVCVVCVCV 605 CCVCCVCCV 602 CVVCCVCVC 551 CVCVVCCVC 543 CVCVCCVCC 533 CCVCVCVVC

# English 9

# $awk '{print $1}' $e|sed -n '/^.\{9\}$/s/[eaiouéÿíáïäóèöîãаüоеиåàê]/V/gp'|sed 's/[rnstlchdmgpbkyfvwzjxqñþçт]/C/g'|sort|uniq -c|sort -nr|head -32

# 3240 CVCCVCCVC 2342 CVCCVCVCV 2295 CVCCVCVCC 1787 CVCVCVCVC 1571 CVCVCCVCC 1454 CVCVCCVCV 1428 CVCVCVCCV 1218 CVCCCVCVC 1156 CVCCVCVVC 1047 CCVCCCVCC 921 CCVCCVCVC 884 CVCCCCVCC 859 VCCVCCVCC 738 CVCCCVCCC 733 CCVCVCVCV 715 CCVCVCVCC 712 CCVCVCCVC 660 CVCVCCVVC 616 CVCCVVCVC 612 VCCVCVCVC 600 CVCVCCCVC 558 CVCCCVCCV 525 CVCVCVCCC 517 CCVVCCVCC 512 CVVCCCVCC 501 VCCVCCVCV 494 CVVCCVCVC 482 CVCCVCCCV 482 CCVCCVCCV 473 CVCCCVVCC 460 CCVCCVCCC 424 CVCCVCCCC

# English 9 (from different dictionary )

# $ cat SixOrMore|sed -n '/^.\{9\}$/s/[aeiouAEIOU]/A/gp'|sed 's/[BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]/C/g;s/A/V/g'|sort|uniq -c|sort -nr|head -32

# 640 CVCCVCVCC 403 CVCCVCCVC 381 CVCVCCVCC 322 CVCCVCVCV 314 CVCVCVCVC 314 CVCCVCVVC 267 VCCVCCVCC 200 CCVCCCVCC 184 CCVCVCVCC 170 CVCVCCVCV 164 CCVCCVCVC 156 VCCVCVCVC 151 CVCCCVCVC 130 CVCCCCVCC 129 CVCVCVCCC 124 CVCVCCVVC 122 VCCVCCVCV 119 CCVVCCVCC 110 CVCCVVCVC 106 CVVCCCVCC 103 CVCVVCVCC 98 CVCVCVCCV 98 CCVCVCCVC 97 CCVCVCVVC 90 CVVCVCVCC 88 VCCVCCVVC 87 CCVCVCVCV 85 VCVCVCVCC 82 VCCCVCVCC 82 CVCVCVVCC 80 VCCCVCCVC 78 CVCCCVCCC

# Note that English (3240 CVCCVCCVC)1 > (1787 CVCVCVCVC)4 (generally consistent across both methods)

# While Spanish (3380 CVCCVCCVC)5 < (6006 CVCVCVCVC)1

**References**

[1] A Chronology of Major Events in the History of Lexicography, from The Oxford Handbook of Lexicography, Edited by Philip Durkin, 2015 seen (January, 2018 at <http://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199691630.001.0001/oxfordhb-9780199691630-miscMatter-8> )

[2] Carter Revard, in Wikipedia. Accessed January 2018 at <https://en.wikipedia.org/wiki/Carter_Revard> .

[3] The Brown Corpus. <https://en.wikipedia.org/wiki/Brown_Corpus>

[4] The History and Philosophy of Project Gutenberg, Michael Hart, 1992 <https://www.gutenberg.org/wiki/Gutenberg:The_History_and_Philosophy_of_Project_Gutenberg_by_Michael_Hart>

[5] The FRELI word list, 2009. Paul M. Hoffman. From http://www.nkuitse.com/freli/  
Creative Commons License (2.0) at freli-20090227/COPYING.

[6] **Anagrams -- words that share letters,  permuted. David Dailey, 2000. Available at** <http://srufaculty.sru.edu/david.dailey/words/anagrams.htm>**.**

**[7] Aegilops, Wikipedia entry, retrived, Feb. 2017.** <https://en.wikipedia.org/wiki/Aegilops>

**[8] TwoOrMore. Words in two or more public access dictionaries, as described at** <https://ello.co/ddailey/post/v7ewsn39omeyt3hbac0_xa> **(Dailey, 2017) in this paper (Section I) and as seen at** <http://cs.sru.edu/~ddailey/cgi/Wotsa?2>

**[9] Roget’s Thesaurus**

**[10] Samuel Johnson’s dictionary.**

**[11] Noah Webster and the American Dictionary. David Micklethwait. 2000.**

**[12] Seven dirty words. From the 1972 monologue by George Carlin. Wikipedia, retrieved Feb. 2017 at** <https://en.wikipedia.org/wiki/Seven_dirty_words>

**[13]** Webster’s Collegiate Dictionary, 1919, By Noah Webster, retrieved at <https://archive.org/details/websterscolle00webs>

**[14] Vocabulary Size, Text Coverage and Word Lists, Paul Nation and Robert Waring, In Schmitt, N. and M. McCarthy (Eds.): Vocabulary: Description, Acquisition and Pedagogy (pp. 6-19). Cambridge: Cambridge University Press. 1997, retrieved at** <http://www.lextutor.ca/research/nation_waring_97.html>

[15] The Moby Project by Grady Ward. As described by Wikipedia, at <https://en.wikipedia.org/wiki/Moby_Project> .

[16] The Project Gutenberg Etext of Moby Word II by Grady Ward. At <http://www.gutenberg.org/files/3201/3201.txt>.

[17] Words\_(Unix), From Wikipedia at https://en.wikipedia.org/wiki/Words\_(Unix)

[18] Development of a Spelling List, M.D. McIlroy. 1982. AT&T Bell Laboratories.  
accessed 2017 at <http://assets.openstudy.com/updates/attachments/4e63a12a0b8b1f45b4ac8b28-orr.andrew-1315343445132-mcilroy_spell_1982.pdf>

[19] SCOWL and friends. From <http://wordlist.aspell.net/>.

[20] Corpus of Contemporary American English. <http://corpus.byu.edu/coca/>.

[21] British National Corpus: What is the BNC? <http://www.natcorp.ox.ac.uk/corpus/index.xml>

[22] List of XML and HTML character entity references, retrieved Feb 2019 from Wikipedia. https://en.wikipedia.org/wiki/List\_of\_XML\_and\_HTML\_character\_entity\_references

[23] Observe, for example, the command and output:

**$ comm <(echo {a..f}|tr ' ' '\n') <(echo {d..h}|tr ' ' '\n')|sed 's/\s//g'|tr '\n' ' ';echo  
a b c d e f g h**

**which forms the union of the set {a..f} with the set {d..h}, namely {a..h}**

**[24]** <https://en.wikipedia.org/wiki/Pneumonoultramicroscopicsilicovolcanoconiosis>

**[25] E-mail not intended to be sent. (Digital archæology). David Dailey on Ello. 2016. https://ello.co/ddailey/post/6lcx4o-ve\_a5c-cj9mr8fa**

### **[26]** Omni-Opticon: a way of visualizing trend-proximities George Shirk IV David Dailey, 2011, **SVG Open 2011, Microsoft NERD, Cambridge MA.**