

Anagram software for cognitive research that enables specification of psycholinguistic variables

ROBERT D. VINCENT, YAEL K. GOLDBERG, and DEBRA A. TITONE
McGill University, Montreal, Quebec, Canada

Anagram tasks are frequently used in cognitive research, and the generation of new scrambled letter combinations is a task well suited to a software solution. Most available programs, however, do not allow experimenters to generate new anagrams flexibly or to characterize existing anagrams using psycholinguistic criteria. They also do not provide detailed information on their source dictionaries. We present anagram software that interfaces with CELEX2, an internationally recognized psycholinguistic database. This software allows users to capitalize on lexical variables and thus enables direct control of psycholinguistic features that may influence the cognitive processes involved in anagram solution.

Anagrams consist of scrambled letter strings that when appropriately arranged create valid lexical forms (e.g., *naagarm* is an anagram of *anagram*). Anagram tasks are frequently used in behavioral research to investigate a wide array of cognitive phenomena. Most prominently, they are used to study the cognitive stages involved in problem solving—specifically, insight or the “aha” phenomenon (Metcalfe, 1986; Novick & Sherman, 2003; Smith & Kounios, 1996; White, 1988)—and to examine how we map orthographic information onto lexical representations during reading (Dominowski & Ekstrand, 1967; Guerrero, 2004; Mayzner & Tresselt, 1958; Novick & Sherman, 2004; Weldon, 1988; White, 1988; Witte & Freund, 2001). Anagram tasks are also used to study meta-cognitive awareness during learning (Kumar, Harizuka, & Koga, 1999), age-related or personality-related changes in cognition (Java, 1992; Papsdorf, Himle, McCann, & Thyer, 1982; Witte & Freund, 1995), memory (Cansino, Ruiz, & Lopez-Alonso, 1999; Java, 1992; Rajaram & Roediger, 1993), and expertise (Novick & Sherman, 2001, 2003). Cognitive neuroscience uses anagram tasks to investigate the neural correlates of these cognitive phenomena (e.g., Chance, Nioka, Sadi, & Li, 2003; Schneider et al., 1996; Sundermeier, 2004). Given the wide use of anagram tasks in cognitive research, methods that facilitate stimulus selection are of great value.

At present, several anagram software programs are available (for a thorough review, see Jordan & Monteiro, 2003). However, their stimulus selection features do not allow experimenters to generate new anagrams flexibly

or to characterize existing anagrams according to certain psycholinguistic criteria. For example, most do not reveal the sources of their vocabulary, although some allow for user-defined vocabularies (Jordan & Monteiro, 2003), and most do not allow batch processing of multiple letter strings. Finally, only one anagram program allows for user-defined constraints on the anagrams generated (Jordan & Monteiro, 2003). Indeed, even some of the most flexible anagram programs that are available do not allow the user to examine the distribution of words that are anagrams of subsequences of the original strings. In our work, we have remedied these limitations by creating anagram software that interfaces with CELEX2 (Baayen, Piepenbrock, & Gulikers, 1995), an internationally recognized psycholinguistic database, which allows the user to capitalize on lexical variables contained therein. Thus, this software allows users to discover new anagrams quickly and to characterize existing anagrams on the basis of psycholinguistic variables that may greatly influence the cognitive processes involved in anagram solution.

Program Design

The source code of the program consists of a few files comprising about 1,200 lines written in the C programming language. Because it is intended to function on a variety of computer system types without requiring modification to the source code, the code makes minimal assumptions about the underlying system and therefore does not include a graphical user interface. Instead, all of the options and parameters can be specified on the command line.

The source code has been compiled and tested successfully on Microsoft Windows, Linux, and Silicon Graphics' IRIX operating systems.

CELEX2 database. The CELEX2 English database consists of 11 tables containing extensive information on frequency, phonology, morphology, orthography, and syntax. CELEX2 is available for purchase from the Linguistic Data Consortium at www ldc.upenn.edu/Catalog/.

We gratefully acknowledge support from the Canada Research Chairs program, NSERC, NARSAD, the Canadian Foundation for Innovation, and the McGill Research Development Fund (to D.A.T.), and to a summer NSERC Fellowship (to Y.K.G.). Address correspondence to R. D. Vincent, who is in the School of Computer Science, or D. A. Titone, who is in the Department of Psychology, McGill University, 1205 Dr. Penfield Ave., Montreal, QC, H3A1B1 Canada (e-mail: bert@cs.mcgill.ca or dtitone@hebb.psych.mcgill.ca).

Our software currently makes use of two tables within CELEX2: the English word form frequency database, which consists of each word form token in the corpus (where, for example, the word forms *work*, *worked*, and *working* are distinct), and the English lemma frequency database, which comprises the orthographic root morphemes (where, for example, *work*, *working*, and *worked* are represented together via their root morpheme “*work*”). The word form database is the more appropriate source for all alternative word forms that may be valid anagram constituents, whereas the lemma database provides unified frequency information that is more salient than frequency information for individual word forms.

These databases are available in a simple text format that can be read and represented in a straightforward way. Each record in the text file corresponds to a single line, and each field is separated by a reverse slash (“\”) character. The word form frequency database contains 160,595 records, and the lemma frequency database contains 52,447 records. These tables are quite large, yet it is well within the capabilities of modern desktop computers to read both tables into memory for rapid access during the anagram evaluation process.

Many of the word forms themselves are compounds that include either a hyphen or a space character. For our purposes, we ignore such entries unless the program user specifically requests that the space or dash be added to the alphabet or interactively enters a string containing a space or dash. Many entries in the database are also capitalized proper nouns or acronyms. These strings are also typically left out of anagram searches, but they can be included through the specification of an option that causes the program to consider lower- and uppercase letters to be equivalent, or by altering the alphabet to specifically include uppercase letters.

The following sections describe some relevant issues pertaining to anagram generation that our software has addressed.

Invariant representation. Most practical anagram generation software must implement an invariant representation of the words such that all anagrammatic words have the same form. This is needed to make rapid comparisons practical. One of the most simple and compact methods is to represent each word by a sorted list of its constituent letters (as suggested in Bentley, 1983). For example, the words *contains* and *sanction* are both represented by the sorted letters *acinnots*, and therefore they are anagrams of one another. Therefore *acinnots* would be the invariant representation of both *contains* and *sanction*. Coincidentally, invariant representations are also used heuristically by competitive Scrabble players to study word lists in preparation for competition (Fatsis, 2002).

Word form and lemma tables. The core of the software is a table consisting of the entire CELEX2 word form frequency database. To facilitate rapid access for searching, the word form table is organized using an “open hash table” approach, wherein each table entry consists of a linked list of word entries, with each list addressed by a hash value. The hash value is calculated by summing the

characters appearing in the invariant form of the word form and taking the modulus relative to the hash table size. This process results in an integer key that is invariant for all words that are constructed from the same list of letters. Each list in the hash table contains all word forms with the same hash value. Although this may include words that are not anagrams of one another, it is guaranteed that all anagrams of a given string (i.e., all words with the same invariant representation) will be stored on the same list in the table. As is shown by the frequency distribution of list lengths in Figure 1, there are typically 10 or fewer entries on each list in the hash table, so this approach permits the software to quickly exclude the vast majority of the word forms from a search.

The CELEX2 lemma frequency database is also loaded in its entirety. Entries in the word form database contain a pointer to the appropriate lemma. This permits rapid retrieval of the full lemma frequency statistics for a given word form.

Modes of operation. The software provides two different modes of operation: interactive and automatic. The interactive mode allows a researcher to provide a hand-generated list of test strings or words for evaluation for possible anagrams. This list can be entered either interactively (i.e., one at a time) or through an input file using standard command line input redirection (e.g., using the “>” character for output or “<” for input). The automatic mode generates a specified number of fixed-length test strings pseudorandomly, possibly given a user-defined alphabet. In either case, the test string is converted to a sorted order and each possible substring is considered. For each substring, all possible anagrams are identified and the lemma frequency information for all orthographically identical word forms is summed and printed. Additional information printed includes the total number of unique word forms matched and the logarithm base 10 of each word form’s lemma frequency. Moreover, given that the vocabulary is based on the CELEX2 database, any additional information about an anagram or its rearranged word forms is readily available for users who have access to this database.

Random number generator. When used in the automatic mode, the program generates and evaluates a large number of pseudorandom strings over a specified alphabet. Both to guarantee identical results across differing computing environments and to avoid the pitfalls associated with many standard random number generators (L’Ecuyer, 2001), we have chosen to incorporate the Mersenne twister pseudorandom number generator (Matsumoto & Nishimura, 1998), which has been found to have excellent distributional properties and an extremely long period.

Search algorithms. When a test string of letters is evaluated for possible anagrams, the program’s normal behavior is to generate and consider all possible subsequences of letters in the test string. The invariant of each subsequence is computed, and the corresponding list is identified in the dictionary table. Anagrams of the subsequence may then be read from the relatively short list

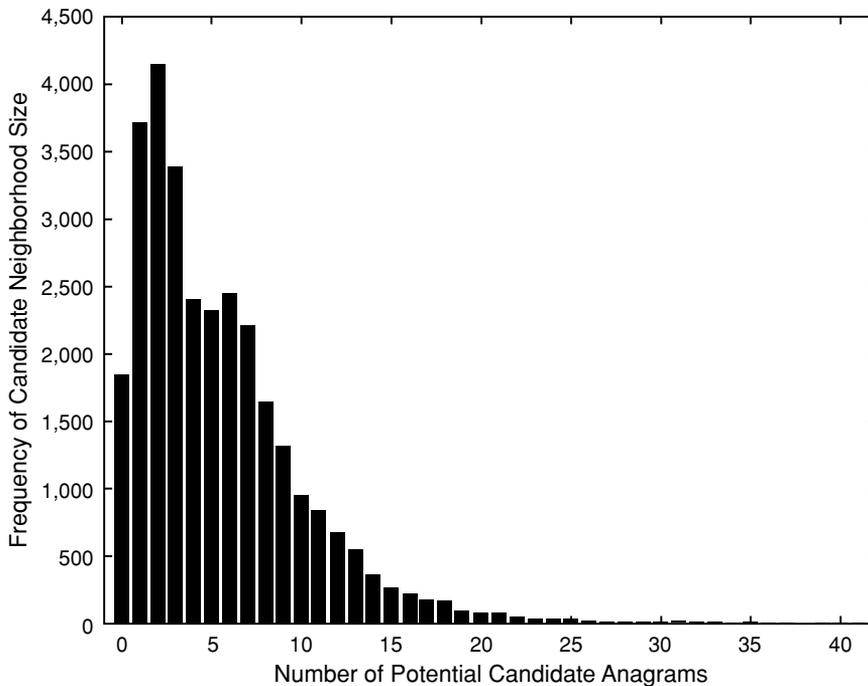


Figure 1. Frequency distribution of hash table list lengths.

of words consistent with this invariant. For typical strings with a length of less than 10, this is considerably faster than exhaustively searching the entire table for potential anagram words. However, for any string of length n , there are 2^n possible unique subsequences. Therefore, as the length of the test string grows, it quickly becomes impractical to generate all subsequences, and above some critical length, it will be more efficient to search the entire dictionary table than to generate subsequences. The program therefore incorporates an alternative look-up algorithm that implements a *brute force* approach, scanning the entire table when considering test strings greater than 16 characters in length. Each word in the dictionary is then considered by evaluating whether or not its invariant representation is a subsequence of the test string's invariant representation. For typical usage with test strings of lengths 5 to 8, the process of generating all possible subsequences for evaluation is more efficient than the brute force search of the entire table.

Program Use

As mentioned above, the program is written with the twin goals of simple use and easy portability to new operating systems. The user interface is therefore a very basic command line with a number of options, summarized in Table 1. The program will print a basic summary of the command line options if any error is detected, or if the user explicitly requests the summary.

Selection of parameters. A variety of parameters can be specified for each run of the program. These parameters include the random number generator's seed value, allowing researchers to duplicate exactly previous

“random” searches where the seed value was noted. Parameters also include the length of random strings, the number of random strings to generate, the minimum (and maximum) word form length to be considered when finding matches, the minimum (and maximum) number of matches required for a random test string to be reported, and options that suppress all random strings that do not contain a specified number of “bingo” solutions—that is, an anagram of the same length as that of the overall test string. It is also possible to restrict the alphabet used for random string generation, to either exclude or emphasize certain letters. The option syntax is similar to that of most UNIX commands, in that options are introduced on the command line with a dash character followed by a single letter identifying the option. Some options require an additional integer or string parameter; others are Boolean flags that do not require any parameter. See Table 1 for a full listing of all the available parameters, their default values, and interpretations.

Automatic mode. In the normal mode of operation, the program generates random strings of a constant length. Each random string is compared against the database and if the string meets the criteria specified, it will be printed along with the associated words discovered within the string. This process continues until the requested number of strings has been discovered and printed.

The randomized nature of the automatic mode provides a relatively fast and unbiased method of discovering novel anagram sequences.

Interactive mode. When the program is run using the “-i” option, the user may specify a list of strings for evaluation. Normally, the program will prompt the user to enter

Table 1
Summary of the Command Line Options

Option Letter	Parameter Type	Description	Default Parameter Value
m	integer	Do not report words of length less than the specified number	1
x	integer	Do not report words of length greater than the specified number	100
s	integer	Generate random strings of length equal to the specified number	8
n	integer	Report exactly this many random strings that meet the rest of the given criteria	10
r	integer	Set random number generator seed to the specified number	100
C	integer	Ignore strings matching fewer than the specified number of words	1
D	integer	Ignore strings matching more than the specified number of words	(unlimited)
B	integer	Ignore strings that contain fewer than the specified number of “bingos”	0
G	integer	Ignore strings that contain more than the specified number of “bingos”	(unlimited)
X	string	Exclude the specified list of letters from the alphabet used to generate random strings	“”
A	string	Use only the specified list of letters when generating random strings	“abcdefghijklmnopqrstuvwxyz”
p	string	Set the path to the CELEX2 database files	“./english”
v	Boolean	Instructs the program to print additional debugging and statistical information	false
c	Boolean	Ignore differences in the case of letters	false
f	Boolean	Use brute force search exclusively	false
i	Boolean	Read the list of strings from the user or input file, instead of generating random strings	false
h	Boolean	Print command line option summary and exit immediately	false

Note—Option characters are case sensitive.

each string interactively from the keyboard; however, it is easy to use standard command line input redirection to cause the program to read a list of strings from a given input file. In this case, the input file should contain exactly one word per line.

This mode is intended to allow easy evaluation of a previously known set of strings, which is especially useful for error checking.

Program output. Program output is normally directed to the user’s command line window. Output can be written to a file of the user’s choice with standard command line output redirection syntax (e.g., “>” or “<”).

The output of the program is designed to be compatible with the commonly used “comma separated values” format, so that it may be easily loaded into standard spreadsheet software for further processing and analysis. Information printed for each test string includes the original test string and the total number of unique word forms matched given the specified selection criteria. For each matched word form found in the test string, the program prints the word form along with its length, the total frequency of all lexically equivalent lemmas per million words in the COBUILD (Collins Birmingham University International Language Database) corpus, and the logarithm base 10

of this frequency. See Figure 2 for a representative segment of this output. Moreover, given that the vocabulary is based on the CELEX2 database, complete additional information about the word forms is readily available for users who have access to this database.

For convenience, the word forms are sorted in order of decreasing length.

Some examples. When the program is run with no arguments, the standard behavior is to generate exactly 10 random strings of length 8. All constituent words will be printed, but only lowercase characters will be considered, so that acronyms and proper nouns will be excluded from the results.

A hypothetical user decides to generate exactly 30 strings of length 6. For purposes of the user’s planned study, it is important that each string can form exactly one bingo solution. The output should be saved in a file called words.txt. In this case, the user could specify the command line:

```
vgt_ana -s6 -n30 -B1 -G1 > words.txt
```

Spacing between the option characters and their arguments is not significant, so the user could equivalently enter

```
Seed,Alphabet
100,abcdefghijklmnopqrstuvwxy
z
vlt,yie,19
,,levity,6,0,0
,,evil,4,58,1.76343
,,levy,4,6,0.778151
,,live,4,531,2.72509
,,tile,4,13,1.11394
,,veil,4,11,1.04139
,,vile,4,4,0.60206
,,yeti,4,0,0
,,ivy,3,5,0.69897
,,lei,3,0,0
,,let,3,431,2.63448
,,ley,3,2,0.30103
,,lie,3,339,2.5302
,,lit,3,92,1.96379
,,lye,3,1,0
,,tie,3,95,1.97772
,,vet,3,9,0.954243
,,vie,3,4,0.60206
,,yet,3,468,2.67025
```

Figure 2. Sample segment of program output with parameters “-s6 -B1 -m3 -n1.”

```
vgt_ana -s 6 -n 30 -B 1 -G 1 > words.txt
```

Suppose a user has a list of strings in a file called inputs.txt. He or she would like to know all anagram words of length greater than 5 that can be generated from each of the input words. The user would enter the following command line:

```
vgt_ana -i -m 5 < inputs.txt
```

Conclusions and Future Work

This software, which is available in source code form at www.psychonomic.org/archive/, greatly simplifies the problem of finding words and anagrams with specific psycholinguistic properties. It provides a fast method of discovering and/or evaluating strings for possible use as stimuli from a very thoroughly studied and documented database of English words. It also allows both information about and control over features such as the number and length of anagram words found in subsequences of the original string. Empirical work with human subjects and stimuli generated by this program is already underway.

Future software extensions will provide control over additional psycholinguistic features such as letter placement restrictions or syntactical information. We plan to implement selection or suppression of potential candidate words on the basis of corpus frequency. Normalization of stimuli with respect to the number of moves required for solution of unique bingo examples will also be considered. We also intend to incorporate bigram frequency calculations directly into this software.

Because of the relatively simple command line operation of the program, it would be straightforward to implement a Web-based interface to facilitate access to the program and database. A user could then visit a URL in the browser, select the desired parameters through a standard HTML form, and optionally enter a list of test strings, and the system could forward the results to the user's email. We hope to implement such a system in the near future for our own use as well as that of other interested researchers.

The authors hope to extend the software to utilize CELEX2 databases in languages other than English (e.g., Dutch and German), and to modify the software to provide similar functionality for alternative linguistic databases.

REFERENCES

- BAAYEN, R. H., PIEPENBROCK, R., & GULIKERS, L. (1995). *The CELEX lexical database* (CD-ROM). Philadelphia: University of Pennsylvania, Linguistic Data Consortium.
- BENTLEY, J. (1983). Programming pearls: Aha algorithms. *Communications of the ACM*, **26**, 623-628.
- CANSINO, S., RUIZ, A., & LOPEZ-ALONSO, V. (1999). What does the brain do while playing Scrabble? ERPs associated with a short-long-term memory task. *International Journal of Psychophysiology*, **31**, 261-274.
- CHANCE, B., NIOKA, S., SADI, S., & LI, C. (2003). Oxygenation and blood concentration changes in human subject prefrontal activation by anagram solutions. *Advances in Experimental Medicine & Biology*, **510**, 397-401.
- DOMINOWSKI, R. L., & EKSTRAND, B. R. (1967). Direct and associative priming in anagram solving. *Journal of Experimental Psychology*, **74**, 84-86.
- FATSI, S. (2002). *Word freak: Heartbreak, triumph, genius, and obsession in the world of competitive Scrabble players*. New York: Penguin.
- GUERRERA, C. (2004). *Flexibility and constraint in lexical access: Explorations in transposed-letter priming*. Doctoral dissertation, University of Arizona.
- JAVA, R. I. (1992). Priming and aging: Evidence of preserved memory function in an anagram solution task. *American Journal of Psychology*, **105**, 541-548.
- JORDAN, T. R., & MONTEIRO, A. (2003). Generating anagrams from multiple core strings employing user-defined vocabularies and orthographic parameters. *Behavior Research Methods, Instruments, & Computers*, **35**, 129-135.
- KUMAR, S., HARIZUKA, S., & KOGA, S. (1999). Interaction of learning awareness and task difficulty in single-solution anagram performance. *Perceptual & Motor Skills*, **88**, 1203-1209.
- L'ECUYER, P. (2001). Software for uniform random number generation: Distinguishing the good and the bad. *Proceedings of the Winter Simulation Conference*, **1**, 95-105.
- MATSUMOTO, M., & NISHIMURA, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling & Computer Simulation*, **8**, 3-30.
- MAYZNER, M. S., & TRESSLET, M. E. (1958). Anagram solution times: A function of letter order and word frequency. *Journal of Experimental Psychology*, **56**, 376-379.
- METCALFE, J. (1986). Premonitions of insight predict impending error. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **12**, 623-634.
- NOVICK, L. R., & SHERMAN, S. J. (2001). *The effects of superficial and structural information on on-line problem solving as a function of expertise*. Unpublished manuscript.
- NOVICK, L. R., & SHERMAN, S. J. (2003). On the nature of insight solutions: Evidence from skill differences in anagram solution. *Quarterly Journal of Experimental Psychology*, **56A**, 351-382.

- NOVICK, L. R., & SHERMAN, S. J. (2004). Type-based bigram frequencies for five-letter words. *Behavior Research Methods, Instruments, & Computers*, **36**, 397-401.
- PAPSDORF, J. D., HIMLE, D. P., MCCANN, B. S., & THYER, B. A. (1982). Anagram solution time and effects of distraction, sex differences, and anxiety. *Perceptual & Motor Skills*, **55**, 215-222.
- RAJARAM, S., & ROEDIGER, H. L., III (1993). Direct comparison of four implicit memory tests. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **19**, 765-776.
- SCHNEIDER, F., GUR, R. E., ALAVI, A., SELIGMAN, M. E. P., MOZLEY, L. H., SMITH, R. J., ET AL. (1996). Cerebral blood flow changes in limbic regions induced by unsolvable anagram tasks. *American Journal of Psychiatry*, **153**, 206-212.
- SMITH, R. W., & KOUNIOS, J. (1996). Sudden insight: All-or-none processing revealed by speed-accuracy decomposition. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **22**, 1443-1462.
- SUNDERMEIER, B. A. (2004). *Priming of semantic relations and anagram solutions: An investigation of cerebral asymmetries, state of arousal and schizotypy*. Doctoral dissertation, University of Minnesota.
- WELDON, M. S. (1988). *Mechanisms underlying data driven retrieval processes*. Doctoral dissertation, Purdue University.
- WHITE, H. (1988). Semantic priming of anagram solutions. *American Journal of Psychology*, **101**, 383-399.
- WITTE, K. L., & FREUND, J. S. (1995). Anagram solution as related to adult age, anagram difficulty, and experience in solving crossword puzzles. *Aging & Cognition*, **2**, 146-155.
- WITTE, K. L., & FREUND, J. S. (2001). Single-letter retrieval cues for anagram solution. *Journal of General Psychology*, **128**, 315-328.

(Manuscript received November 28, 2005;
revision accepted for publication March 2, 2006.)