Behavior Research Methods
2010, 42 (3), 627-633
doi:10.3758/BRM.42.3.627

Wuggy: A multilingual pseudoword generator

Emmanuel Keuleers and Marc Brysbaert
Ghent University, Ghent, Belgium

Pseudowords play an important role in psycholinguistic experiments, either because they are required for performing tasks, such as lexical decision, or because they are the main focus of interest, such as in nonword-reading and nonce-inflection studies. We present a pseudoword generator that improves on current methods. It allows for the generation of written polysyllabic pseudowords that obey a given language's phonotactic constraints. Given a word or nonword template, the algorithm can quickly generate pseudowords that match the template in subsyllabic structure and transition frequencies without having to search through a list with all possible candidates. Currently, the program is available for Dutch, English, German, French, Spanish, Serbian, and Basque, and, with little effort, it can be expanded to other languages.

Nonwords are essential in lexical decision tasks in which participants are confronted with strings of letters or sounds and have to decide whether the stimulus forms an existing word. Together with word naming, semantic classification, perceptual identification, and eye-movement tracking during reading, the lexical decision task is one of the core instruments in the psycholinguist's toolbox for the study of word processing.

Although researchers are concerned particularly with the quality of their word stimuli (because their investigation depends on them), there is plenty of evidence that the nature of the nonwords also has a strong impact on lexical decision performance. As a rule, the more dissimilar the nonwords are to the words, the faster are the lexical decision times and the smaller is the impact of word features such as word frequency, age of acquisition, and spelling–sound consistency (e.g., Borowsky & Masson, 1996; Gerhand & Barry, 1999; Ghyselinck, Lewis, & Brysbaert, 2004; Gibbs & Van Orden, 1998). For instance, in Gibbs and Van Orden (Experiment 1), lexical decision times to the words were shortest (496 msec) when the nonwords were illegal letter strings (i.e., letter sequences, such as *ldfa*, that are not observed in the language), longer (558 msec) when the nonwords were legal letter strings (e.g., *dilt*), and still longer (698 msec) when the nonwords were pseudohomophones (i.e., sounding like real words, e.g., *durt*). At the same time, the difference in reaction times (RTs) between words with a consistent rhyme pronunciation (e.g., *beech*) and matched words with an inconsistent rhyme pronunciation (e.g., *beard* [inconsistent with *heard*]) increased. Because of the impact of the nonwords on lexical decision performance, there is general agreement among researchers that nonwords should be legal nonwords, unless there are theoretical reasons to use illegal nonwords. Legal nonwords that conform to the orthographic and phonological patterns of a language are also called *pseudowords*.

Although the requirement of pseudowords solves many problems for the creation of nonwords in the lexical decision task, there are additional considerations that must be taken into account. Because lexical decision is, in essence, a signal detection task (e.g., Ratcliff, Gomez, & McKoon, 2004), participants in a lexical decision task not only base their decision on whether the stimuli belong to the language, they also rely on other cues that help to differentiate between the word and nonword stimuli. In the same way that participants learn ties in apparently random materials generated on the basis of an underlying grammar (i.e., the phenomenon of implicit learning; Reber, 1989), so are participants susceptible to systematic differences between the word trials (requiring a "yes" response) and the nonword trials (requiring a "no" response). They exploit these biases to optimize their responses. Chumbley and Balota's (1984) study provides an example of this process. Because of an oversight, in their Experiment 2, the nonwords were on average one letter shorter than were the words (stimuli ranged from three to nine letters). This gave rise to rather fast RTs (566 msec) and small effects of the word variables under investigation. When Chumbley and Balota (Experiment 3) repeated the experiment with proper nonwords, RTs became longer (579 msec) and the effects became stronger. Another example of a subtle bias in lexical decision tasks was reported by Rastle and Brysbaert (2006). They reviewed the literature on the masked phonological priming effect, where it has been shown that a target word is recognized faster when it is preceded by a pseudohomophonic prime than when an orthographic control is presented. The target word FARM is responded to faster in a lexical decision task when it is preceded by the masked prime *pharm* than when it is preceded by the control prime *gharm*. However, Rastle and Brysbaert noticed that, in these experiments, every time the prime was a pseudohomophone, it was followed by a word (i.e.,

E. Keuleers, emmanuel.keuleers@ugent.be

the target that sounds like the pseudohomophone). When Rastle and Brysbaert corrected for this confound, they observed that the phonological priming effect decreased from 13 to 9 msec.

For the above reasons, researchers have to be very careful in the design of nonwords. They must make sure that there are no systematic differences between the words and the nonwords, other than the fact that the former belong to the language and the latter do not (see Rastle, Harrington, & Coltheart, 2002, for a similar message). This requirement is particularly relevant when the number of trials is large and participants have the time to tune in to any bias in the stimulus materials. For instance, if many more nonwords than words end with the letters *-ck*, participants are likely to pick up this correlation and, after some time, will show faster rejection times for nonwords ending with *-ck* and slower acceptance times for words ending with *-ck*.

Current Options for Making Pseudowords

A review of the literature suggests that researchers have been using two methods to create pseudowords. The dominant procedure is to start from the word stimuli in the experiment and to change one or more letters in these words to turn them into pseudowords. For instance, the word *milk* can be changed into a nonword by changing any single letter. Hence, we could get nonwords like *pilk*, *malk*, *mirk*, or *milp*. In this procedure, the researcher's judgment is the primary criterion to evaluate the goodness of the pseudowords. This judgment, in turn, relies on the constraints picked up by the researcher from the language (e.g., the observation that English monosyllabic words can start with the letters *pi-* and *ma-* and can end with the letters *-rk* and *-lp*). Arguably, the largest experiment in which this approach was used is the English Lexicon Project (Balota et al., 2007), in which the researchers created over 40,000 pseudowords by changing one or two letters in the word stimuli.

The second approach is used by programs such as WordGen (Duyck, Desmet, Verbeke, & Brysbaert, 2004), which is available for English, Dutch, German, and French, and MCWord (Medler & Binder, 2005), which is available only for English. These programs allow the user to generate a number of pseudowords by stringing together high-frequency bigrams or trigrams and to compute statistics that help the user to select the pseudoword that best matches a given word on a number of criteria. Such a criterion could be the number of words that can be made by changing a letter (the so-called *orthographic neighbors*). For instance, four well-known and four less familiar English words can be made by changing one letter of the word *milk* (*silk*, *mild*, *mile*, *mink*, *mill*, *bilk*, *mick*, and *milt*). So, to match the word *milk*, we would look for a nonword that has the same number of orthographic neighbors. Another criterion could be the frequencies of the successive letter pairs in the word (*-m*, *mi*, *il*, *lk*, *k-*). Then, we would try to match the pseudoword on these frequencies (this is the so-called *bigram frequency criterion*; sometimes researchers also control for *trigram frequencies*—i.e., the frequencies of three-letter sequences). WordGen, for instance, can inform the user about the number of neighbors a word or a nonword has and what its summed bigram frequency is. It would inform the user that the word *milk* has 8 neighbors and has a summed bigram frequency of 3,582 and that the pseudowords score as follows: *pilk* (7 neighbors, summed bigram frequency = 3,183), *malk* (12 neighbors, summed bigram frequency = 6,329), *mirk* (9 neighbors, summed bigram frequency = 2,949), and *milp* (5 neighbors, summed bigram frequency = 3,497). It would also tell an informed user[1] that, on the two criteria, the pseudoword *filk* may be a better option than is *pilk*, because it has 8 neighbors and a summed frequency of 3,083.

Another way of searching for pseudowords that match a given word is the ARC nonword database (Rastle et al., 2002). This database contains all legal monosyllabic English nonwords with various features (e.g., bigram frequency, trigram frequency, pronunciation, whether or not the nonword is a pseudohomophone, the consistency of the rhyme pronunciation). Here again, the user can search for the pseudoword in the list that best matches the word on specified criteria.

Limitations of the Available Solutions

A major limitation of the subjective judgment strategy is that the outcome is likely to depend on the judge's experience with the language and with nonwords. This disadvantages young researchers and researchers who do not fully master the language (e.g., nonnative English speakers doing research in English). It also introduces the possibility of experimenter biases, because researchers may have an idiosyncratic preference to change certain letters or letter combinations. It further makes it difficult to equate the "wordlikeness" of nonwords of different length. For instance, if only one letter is changed to make a nonword, the nonword increasingly resembles the word as the latter becomes longer (compare *fand/fund* to *fandament/fundament*).

The availability of criteria such as the number of neighbors or the summed bigram frequency is a big help for the researcher. However, at present, this information is largely limited to short words. The ARC nonword database provides only information for monosyllabic nonwords, and the time needed to generate nonwords with WordGen increases rapidly with the length of the nonword, because the software does not allow researchers to systematically search the problem space. For instance, the best search strategy to find good nonwords for *milk* is to start by generating many English nonwords with from seven to nine neighbors, summed bigram frequencies between 3,000 and 4,000, and the letter patterns **ilk*, *m*lk*, *mi*k*, and *mil**. The latter cannot be done in a single search but requires the researcher to run four searches. In addition, the algorithm does not search systematically and, in a sparse region, is likely to come up with the same solution over and over again, even though another solution may be available (a way around this is to have many nonwords generated and to check whether all are the same).

Because of these problems, and because we had to create tens of thousands of mono- and disyllabic nonwords for a number of studies we wanted to run, we decided to build a more sophisticated algorithm. Because the purpose was to collect data in different languages, we wanted the algorithm to be applicable to any alphabetic language.

THE WUGGY ALGORITHM[2]

The traditional method to generate pseudowords, as was used to fill the ARC nonword database (Rastle et al., 2002), is based on combining subsyllabic elements that are legal in the language of choice. A conventional way to describe a syllable is to divide it into onset, nucleus, and coda. The element of the syllable that has maximal sonority is called the *nucleus*. In most cases, this is a vowel, although in some languages a consonant with high sonority, such as *r*, can also be the nucleus, as in the Serbian word *crn* ("black"). The nucleus is an essential element of every syllable and optionally can be preceded as well as followed by consonants; these are called, respectively, the *onset* (the consonants before the nucleus) and the *coda* (the consonants after the nucleus). For instance, by combining the legal onset *b* (as in *bat*) with a legal nucleus *u* (as in *fun*) and with a legal coda *p* (as in *ship*), we get the pseudoword *bup*, which is phonotactically legal in English. The major disadvantage to this approach is that it leads to a combinatorial explosion. For monosyllabic words, the list is still manageable (hundreds of thousands of pseudowords), but combining elements into polysyllabic strings quickly leads to billions of phonotactically legal possibilities. Finding a pseudoword matching some specific constraints soon becomes unfeasible, because there are too many candidates to search.

The Wuggy algorithm resolves this problem by building a grammar of the lexicon as a bigram[3] chain: (1) To build the bigram chain, a list of syllabified words in a particular language is required. (2) The algorithm segments each word in this list into subsyllabic elements. (3) From each subsyllabic element, a tuple is constructed, consisting of four components: the letters of the subsyllabic element, the position of the element in the word, the number of elements in the word from which it originates, and the next subsyllabic element. (4) Then, there is a lookup to see whether a *link* consisting of the first three components already exists in the bigram chain. (5) If the link does not yet exist, it is inserted and the next subsyllabic element is added as a possible continuation. (6) If it does exist, its frequency is updated and, if necessary, the next subsyllabic element is added to the possible continuations for that link. (7) When all words in the list have been processed, the bigram chain constitutes an inductive phonotactic grammar of the language. (8) By recursively iterating through the chain, we can generate all possible words and pseudowords.

The algorithm has the built-in restriction that, to generate sequences of *n* syllables, only elements originating from words with *n* syllables are used, as if there were separate grammars for words with different numbers of syllables. This is a careful consideration, based on the facts that, for instance, the first syllable of a disyllabic word differs in many respects from the second syllable and that both differ from monosyllabic words (e.g., the latter are often longer). Therefore, we used position-dependent syllables (e.g., monosyllabic pseudowords are generated on the basis of monosyllabic words, and disyllabic pseudowords are generated on the basis of disyllabic words).

To output orthographic pseudowords, Wuggy is supplied with a list containing the syllabified orthography of each word. At first sight, it may seem odd that no phonetic representations are used. For the ARC database, for instance, orthographic pseudowords were made by first generating phonetic pseudowords and then transcribing them using phoneme-to-grapheme conversion rules. Wuggy does not use phonetic representations, but it uses a list of possible syllable nuclei for each particular language to directly segment spelled syllables into orthographic subsyllabic elements. Although there is no principled way to resolve all ambiguities in segmenting spelled words, this does not often lead to problems when generating spelled pseudowords. Take, for instance, the words *house* and *touch*. The status of *u* is ambiguous, because, in the spoken syllable /haʊs/, it can be treated as the consonant /ʊ/, which is part of the coda, whereas in the spoken syllable /tʌtʃ/ it is part of the nucleus /ʌ/. In Wuggy's English language module, *ou* is considered a possible nucleus. Therefore, *house* and *touch* are segmented as *h-ou-se and t-ou-ch*. Because Wuggy strings together two segments if they are found to occur in sequence in some word in the lexicon, we will get the pseudowords *houch* (*h-ou-ch*) and *touse* (*t-ou-se*). Although the pronunciation of these pseudowords is unclear, none of the possible pronunciations violate the phonotactic constraints of English. For our purposes (i.e., the generation of spelled pseudowords), this approach seems sufficient. Of course, the quality of the pseudowords that are generated also depends on the correct syllabification of the words that Wuggy uses to construct its model from. We hope that users will give feedback about cases in which the syllabification seems to be unsatisfactory or in which the segmentation rules give unexpected results, so that this can be improved in subsequent versions.

A limitation of the Wuggy algorithm is that it does not generate the pronunciations for orthographic pseudowords. This means that Wuggy cannot indicate whether a word is a pseudohomophone. A solution to this problem would be to add individual grapheme-to-phoneme conversion modules for each language, which is beyond the scope of the program in its current state.

Up to now, we have discussed how Wuggy constructs a model that allows it to generate all possible pseudowords. However, because billions of polysyllabic pseudowords can be generated, such a list would not be searchable within a reasonable time. We resolved this problem by observing that, in psycholinguistic research, usually an existing word or stimulus is used as a template for a pseudoword stimulus to be generated. And, in the case that pseudowords are required that specifically do not resemble a certain template, another template usually can be specified. Therefore, the bigram chain can be restricted to generate only words matching the template to a particular degree, by removing all elements of the chain that do not match the restrictions. Currently, the bigram chain can be restricted in two ways. The first is the *segment length criterion*. A template such as *bridge* can be seen as a sequence of subsyllabic elements *br-i-dge*, with lengths 2–1–3. If we keep only the elements of the bigram chain that have the same length at the same position, the number of words that can be generated is much smaller, and the resulting pseudowords will have exactly the same subsyllabic structure as does the template. The second way in which we can restrict the number of words that can be generated is by using a frequency criterion. If the bigrams [_,*br*], [*br*,*i*], [*i*,*dge*], and [*dge*,_] occur with frequencies 125, 25, 4, and 29, respectively, we can filter out all links that do not occur within a given deviation of this particular frequency. This restriction makes the Wuggy algorithm particularly effective, because it is initially set to a very small value (2 above and below the reference frequencies), which dramatically reduces the number of words that can be generated. If this restriction does not result in enough candidates, a less severe restriction is applied (the next power of 2), and so on. We call this method of generating sequences with matching frequencies *concentric search.*

The concentric search mode turns out to have two other advantages. First, in the vast majority of cases, the changes involve two subsyllabic elements that have a low transition frequency (the number of words in which two specific subsyllabic elements occur in sequence). These are easier to replace than are word segments with high transition frequencies. As an example, a monosyllabic word ending in -*s* will virtually always result in a nonword ending in -*s*, because there is no replacement of this letter that does not involve a massive change of transition frequency (given that so many words end in -*s*). In other words, the algorithm tends to go for the weakest link in the word. For the same reason, words with frequent syllables (e.g., prefixes) tend to keep that syllable, because it cannot be changed without introducing a major shift in transition frequency. Second, because the frequency differences are kept as minimal as possible, the algorithm usually replaces high-frequency segments by other high-frequency segments and replaces low-frequency segments by other low-frequency segments.

When the segment length restriction and the concentric search mode are used together, the Wuggy algorithm can often immediately generate pseudowords matching a given template in transition frequency and subsyllabic structure.

The default option in Wuggy is to generate pseudowords that differ from the template in one out of three segments, where onset and coda are always counted as a segment, even if they are empty (e.g., *at* has an empty onset; *pro* has an empty coda). Thus, in a monosyllabic word, either the onset, the nucleus, or the coda would be changed. In a disyllabic word, two segments would be changed. In the latter case, the algorithm does not require the changes to be in two different syllables, because such a constraint usually involves higher frequency deviations from the template. The default option is, thus, to make as many changes as there are syllables, although this does not have to result in exactly one change in every syllable.

To make the operation of the algorithm more concrete, we will discuss a few examples. First, the best nonwords for *milk*, according to the Wuggy algorithm, are *misk* and *mirl*. The transition frequencies between -*i*- and -*sk* or -*rl* are almost the same as the one between -*i*- and -*lk* (a difference of 1 in favor of -*sk* and -*rl*). Of the previously generated nonwords, the best matching is *mirk*. The end letters -*irk* occur in 13 more monosyllabic words in the corpus than do the end letters -*ilk*. The transition frequencies are also higher for *malk* (*ma*- occurs in 34 more monosyllabic words than does *mi*-) and *pilk* (44 more monosyllabic words start with *pi*- than with *mi*-). Finally, the nonword *milp* is not produced by Wuggy, because the end sequence -*ilp* never occurs in English monosyllabic words. In conclusion, of the 4 nonwords we made on the basis of sound judgment, 3 were too good (i.e., were more wordlike than the word itself on the transition frequency criterion) and 1 was rather bad (the end sequence -*ilp* never occurs in English words).

To illustrate the Wuggy output for a wider range of words, we collected the best pseudoword matches with default parameter settings for the English sentence "This sentence has been modified by the Wuggy algorithm." This gave the output "Thas muntence mas boan setified py thi Giggy alworyard."

Because the Wuggy algorithm is generic, it can be used for all languages that have an alphabetic script. As soon as the program has a list of syllabified words and is informed about how the syllables are segmented, it can operate.

Table 1 lists the modules for generating orthographic pseudowords available at the time of writing. Although researchers with programming skills may be happy to use the source code of the algorithm, we decided to write an interface that makes the algorithm easy to use for everyone. In addition, we added a few options so that researchers are not bound to the choices we made for our research.

Table 1
Subsyllabic Modules for Generating Orthographic Pseudowords

Language	Lexicon	Source
Basque	18,486 Basque word forms from E-HITZ	Perea et al. (2006)
Dutch	293,749 Dutch word forms from the CELEX lexical database	Baayen, Piepenbrock, & Gulikers (1995)
English	66,330 English word forms from the CELEX lexical database	Baayen et al. (1995)
French	116,194 French word forms from the Lexique 3 database	New, Pallier, Brysbaert, & Ferrand (2004)
German	236,890 German word forms from the CELEX lexical database	Baayen et al. (1995)
Serbian (Latin and Cyrillic)	144,105 word forms from the frequency dictionary of contemporary Serbian language	Kostić (1999)
Spanish	31,490 Spanish word forms from the base-lexicon of B-PAL	Davis & Perea (2005)

Downloading and Installing

Wuggy is available for Macintosh, Windows, and Linux operating systems at http://crr.ugent.be/Wuggy/. To install Wuggy on a computer running Mac OS X, the Wuggy[version].dmg file must be downloaded. Next, the folder "Wuggy app" must be dragged to the Applications folder, and the "Wuggy" folder must be dragged to the Applications Support folder. To install Wuggy on Windows, the Wuggy-[*version*]-setup.exe executable must be downloaded. This opens a wizard that installs Wuggy. Linux users can download the source files and start the application from the command line.

Overview of Operation

Wuggy has a native look and feel on the different platforms (Mac OS X, Windows, Linux). Figure 1 shows Wuggy's main window on OS X. After starting the program, a language module should be chosen from the "General Settings" options on the right. This loads a syllabified language lexicon, which allows the program to compute the model for the language. The lexicon is also used to syllabify input and to test the lexicality of generated forms. Loading a language module may take a few minutes on older computers. In Figure 1, the English language module is loaded.

Then, reference words can be input by typing them in the appropriate column or reading them from a file. In Figure 1, the words *milk* and *sentence* have been input and then syllabified by choosing the Syllabify option from the Tools drop-down menu.

When input is given, the program is ready to generate candidates. The default values for pseudoword generation are those that we found most appropriate for our research. By default, Wuggy outputs only pseudowords and searches either for up to 10 sec or until 10 candidates are generated. Additionally, the candidates are required to match the subsyllabic structure of the input word, to have the same length (in letters) as the input word, to have the fewest possible deviations in transition probabilities from the input word, and to match two out of three subsyllabic segments.

Choosing the Run option from the Generate drop-down menu opens the Results window. Figure 2 shows the output for the words *milk* and *sentence* using the default output restrictions and with all output options checked.

Overview of Options

Main window. First column (Word): Reference words can be entered manually or read from a text file by selecting the Open Input Sequences option from the File drop-down menu. The input file must be in tab-delimited format. To ensure maximal flexibility and compatibility, Wuggy reads Unicode (UTF-8) encoded files.

Second column (Syllables): Wuggy automatically syllabifies all words it finds in its lexicon. Choosing the Syllabify option from the Tools drop-down menu fills the second column with the syllabified versions of the input in the first column. For input words that are not found in the lexicon, a syllabified version should be entered manually.

Third column (Matching Expression): Typing a regular expression here requires all generated pseudowords to match that regular expression. For instance, if only pseudowords ending in *-ing* are required, one would type *.+ing$* in this column. Information about regular expressions is widely available online (e.g., http://en.wikipedia.org/wiki/Regular_expression, accessed on December 12, 2009).

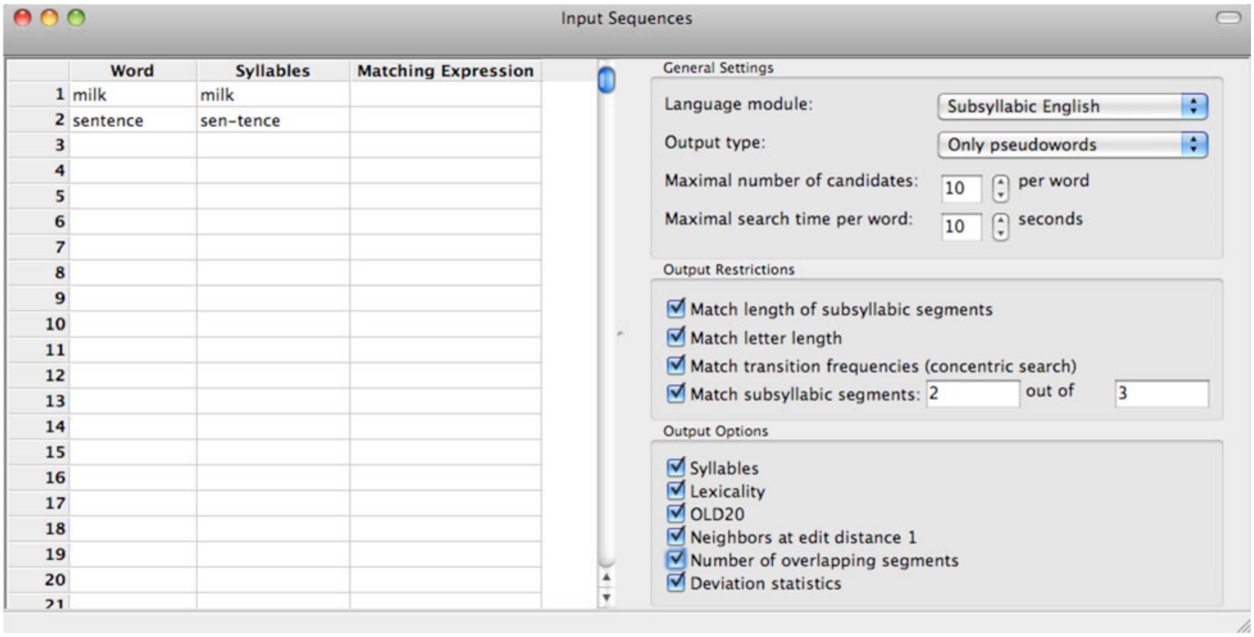

Figure 1. Main window of the application.

Results

	Word	Match	Lexicality	Old20	Old20_Diff	Ned1	Ned1_Diff	Overlap_Ratio	Maxdeviation	Summed_Deviation	Maxdeviation_Transition
1	milk	misk	N	1.6	0.25	8	-3	2/3	1	1	_mi[sk_]
2	milk	mirl	N	1.75	0.4	5	-6	2/3	1	2	_m[irl]_
3	milk	molk	N	1.65	0.3	7	-4	2/3	-3	3	_m[olk]_
4	milk	mirm	N	1.85	0.5	3	-8	2/3	3	5	_mi[rm_]
5	milk	migh	N	1.8	0.45	4	-7	2/3	3	3	_mi[gh_]
6	milk	mibe	N	1.6	0.25	8	-3	2/3	-3	3	_mi[be_]
7	milk	mimb	N	1.9	0.55	2	-9	2/3	-3	5	_mi[mb_]
8	milk	mirp	N	1.9	0.55	2	-9	2/3	-8	11	_mi[rp_]
9	milk	mife	N	1.4	0.05	12	1	2/3	-8	10	_mi[fe_]
10	milk	misp	N	1.8	0.45	4	-7	2/3	-5	6	_mi[sp_]
11	sen-tence	mun-tence	N	3.2	0.8	0	-2	2/3	-28	40	_[mu]ntence_
12	sen-tence	sis-tence	N	2.7	0.3	0	-2	2/3	29	60	_[si]stence_
13	sen-tence	run-tence	N	3.25	0.85	0	-2	2/3	55	108	[_r]untence_
14	sen-tence	ron-tence	N	2.95	0.55	0	-2	2/3	-63	166	_[ro]ntence_
15	sen-tence	men-tette	N	3.4	1.0	0	-2	2/3	-45	98	_[me]ntette_
16	sen-tence	men-tells	N	2.85	0.45	0	-2	2/3	-45	66	_[me]ntells_
17	sen-tence	men-telts	N	2.95	0.55	0	-2	2/3	-61	138	_mente[lts_]
18	sen-tence	men-tects	N	2.95	0.55	0	-2	2/3	-45	66	_[me]ntects_
19	sen-tence	men-teres	N	2.45	0.05	0	-2	2/3	48	121	_mente[res_]
20	sen-tence	men-terns	N	2.45	0.05	0	-2	2/3	-45	93	_[me]nterns_

Figure 2. Output window with results for the words *milk* and *sentence*.

General settings. Language module: Currently, there are language modules available for Basque, Dutch, English, French, German, Serbian, and Spanish.

Output type: This option determines whether Wuggy outputs only pseudowords, only words, or both. Choosing "word" makes Wuggy find the closest word neighbors of a target word.

Maximal number of candidates: The maximum number of candidates to be generated for each word.

Maximal search time per word: The maximal time that to be spent on trying to find candidates.

Output restrictions. Match length of subsyllabic segments: Checking this option causes only candidates with the same subsyllabic structure as the input word to be output. This option speeds up the output, because there are fewer candidates to consider.

Match letter length: Checking this option generates candidates with the same number of letters as the input word. This option is redundant if the option "Match length of subsyllabic segments" is checked.

Match transition frequencies (concentric search): This option operates the concentric search algorithm as described above. First, the algorithm tries to generate candidates that exactly match the transition frequencies of the reference word. Then, the maximal allowed deviation in transition frequencies increases by powers of 2 (i.e., ± 2, ± 4, ± 8, etc.). Not checking this option results in the generation of pseudowords without consideration for transition frequencies. However, because the problem space is less well defined in that case, it may take longer.

Match subsyllabic segments: Here, a particular ratio of overlapping segments can be specified. The default value (2/3) generates candidates that are very wordlike but are not easily identifiable as related to an existing word.

Output options. Syllables: This will give syllabified output. Unchecking this option will give plain strings.

Lexicality: Indicates whether the generated form is a word (W) or a nonword (N). This is particularly useful with the "Output Type > Both" option in the General Settings.

OLD20: Checking this option computes the average orthographic Levenshtein distance between the generated candidate and its 20 most similar words in the lexicon. This gives a good indication of the neighborhood size and density of the nonword (Yarkoni, Balota, & Yap, 2008). A small value of OLD20 indicates that many words can be made by changing a single letter (by substitution, deletion, or insertion). The difference in OLD20 between the generated nonword and the reference word is also shown. Lower values indicate that the candidate has a denser neighborhood. Setting this option slows down Wuggy considerably.

Neighbors at edit distance 1: This option outputs the number of orthographic neighbors at edit distance 1. This is the number of words that can be made from the candidate by substituting, deleting, or inserting a single letter. Setting this option slows down the program considerably.

Figure 2 shows the output when both OLD20 and Neighbors at edit distance 1 have been selected for the target word *milk*. This output clearly shows that all but 1 of the proposed nonwords have fewer neighbors than does the target word *milk*. For instance, *misk* has 8 neighbors of edit distance 1, which is 3 fewer than *milk*. Similarly, the average edit distance to the 20 closest neighbors is 1.6, which is 0.25 more than that to *milk*. *Mife* looks like a better choice than *misk*, because it has 1 neighbor more at edit distance 1 than does *milk*, rather than 3 fewer. Given that OLD20 is an important variable in lexical decision RTs (Yarkoni et al., 2008), researchers may prefer to keep this as close to the word value as possible, as long as it does not change the difference in transition frequency too much. This shows the advantage of having more than 1 candidate proposed by Wuggy.

Number of overlapping segments: With this option checked, the number of segments that overlap in the generated sequence and the reference sequence is shown as a fraction.

Deviation statistics: This option shows the largest difference in transition frequencies between the subsyllabic segments in the generated sequence and those in the reference sequence. For instance, if this measure is 14, the generated sequence contains a transition that occurs in 14 more words than does the equivalent transition in the reference sequence. The frequencies of all other transitions are closer to the frequencies of the transitions in the reference sequence. Checking this option also outputs the sum of all transition frequency deviations (absolute values) and a column showing where in the string the maximally deviating transition is situated.

CONCLUSION

We have written a computer program that allows researchers to find the best matching pseudowords in terms of subsyllabic structure and transition frequencies between subsyllabic elements. This algorithm and its associated user interface are likely to improve the quality of the nonwords used in lexical decision tasks and other psycholinguistic experiments. The procedure computes matching nonwords in very little time and is limited only by its input lexica where length is considered. Finally, the algorithm can easily be extended to new languages.

AUTHOR NOTE

Address correspondence to E. Keuleers, Department of Experimental Psychology, Ghent University, Henri Dunantlaan 2, B-9000, Ghent, Belgium (e-mail: emmanuel.keuleers@ugent.be).

REFERENCES

Baayen, R. H., Piepenbrock, R., & Gulikers, L. (1995). The CELEX lexical database (release 2) [CD-ROM]. Philadelphia: Linguistic Data Consortium, University of Pennsylvania.

Balota, D. A., Yap, M. J., Cortese, M. J., Hutchison, K. A., Kessler, B., Loftis, B., et al. (2007). The English Lexicon Project. *Behavior Research Methods*, **39**, 445-459.

Berko, J. (1958). The child's learning of English morphology. *Word*, **14**, 150-177.

Borowsky, R., & Masson, M. E. J. (1996). Semantic ambiguity effects in word identification. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **22**, 63-85. doi:10.1037/0278-7393.22.1.63

Chumbley, J. I., & Balota, D. A. (1984). A word's meaning affects the decision in lexical decision. *Memory & Cognition*, **12**, 590-606.

Davis, C. J., & Perea, M. (2005). BuscaPalabras: A program for deriving orthographic and phonological neighborhood statistics and other psycholinguistic indices in Spanish. *Behavior Research Methods*, **37**, 665-671.

Duyck, W., Desmet, T., Verbeke, L. P. C., & Brysbaert, M. (2004). WordGen: A tool for word selection and nonword generation in Dutch, English, German, and French. *Behavior Research Methods, Instruments, & Computers*, **36**, 488-499.

Gerhand, S., & Barry, C. (1999). Age-of-acquisition and frequency effects in speeded word naming. *Cognition*, **73**, B27-B36.

Ghyselinck, M., Lewis, M. B., & Brysbaert, M. (2004). Age of acquisition and the cumulative-frequency hypothesis: A review of the literature and a new multi-task investigation. *Acta Psychologica*, **115**, 43-67.

Gibbs, P., & Van Orden, G. C. (1998). Pathway selection's utility for control of word recognition. *Journal of Experimental Psychology: Human Perception & Performance*, **24**, 1162-1187.

Kostić, Đ. (1999). Frekvencijski rěcnik savremenog srpskog jezika [Frequency dictionary of contemporary Serbian language]. Yugoslavia: University of Belgrade, Institute for Experimental Phonetics and Speech Pathology and Laboratory for Experimental Psychology.

Medler, D. A., & Binder, J. R. (2005). MCWord: An on-line orthographic database of the English language. Available at www.neuro.mcw.edu/mcword/.

New, B., Pallier, C., Brysbaert, M., & Ferrand, L. (2004). *Lexique 2*: A new French lexical database. *Behavior Research Methods, Instruments, & Computers*, **36**, 516-524.

Perea, M., Urkia, M., Davis, C. J., Agirre, A., Laseka, E., & Carreiras, M. (2006). E-Hitz: A word frequency list and a program for deriving psycholinguistic statistics in an agglutinative language (Basque). *Behavior Research Methods*, **38**, 610-615.

Rastle, K., & Brysbaert, M. (2006). Masked phonological priming effects in English: Are they real? Do they matter? *Cognitive Psychology*, **53**, 97-145.

Rastle, K., Harrington, J., & Coltheart, M. (2002). 358,534 nonwords: The ARC nonword database. *Quarterly Journal of Experimental Psychology*, **55A**, 1339-1362.

Ratcliff, R., Gomez, P., & McKoon, G. (2004). A diffusion model account of the lexical decision task. *Psychological Review*, **111**, 159-182.

Reber, A. S. (1989). Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*, **118**, 219-235.

Yarkoni, T., Balota, D., & Yap, M. (2008). Moving beyond Coltheart's *N*: A new measure of orthographic similarity. *Psychonomic Bulletin & Review*, **15**, 971-979.

NOTES

1. This information is not given at once. One has to search for four-letter nonwords with eight neighbors, with a summed bigram frequency between 3,000 and 4,000, and ending with the letters -*ilk*.

2. The program is called *Wuggy* in honor of one of the first studies involving nonwords. In this study, Berko (1958) presented children with a picture of a birdlike figure and told them "This is a WUG." Subsequently, the children saw a picture with two such figures and were told "Now there is another one. There are two of them. There are two ___." This test is known in the literature as the WUG Test.

3. To avoid confusion, it is important to note that *bigram* is used in its generic sense (a sub-sequence of two items from a given sequence). In this context, *bigram* refers to a sequence of two subsyllabic elements.

(Manuscript received December 17, 2009;
revision accepted for publication March 14, 2010.)