UniPseudo: A universal pseudoword generator

Boris New, Jessica Bourgin, Julien Barra, Christophe Pallier

UniPseudo : A Universal Pseudoword Generator

Boris New[1], Jessica Bourgin[1], Julien Barra[1], & Christophe Pallier[2]

[1] University Savoie Mont Blanc (USMB), Chambéry, France, Laboratoire Psychologie et Neurocognition, CNRS UMR 5105.
[2] Cognitive Neuroimaging Unit, INSERM, CEA, CNRS, Université Paris-Saclay, NeuroSpin center, 91191 Gif/Yvette, France

Abstract
Pseudowords are letter strings that look like words but are not words. They are used in psycholinguistic research, particularly in tasks such as lexical decision. In this context, it is essential that the pseudowords respect the orthographic statistics of the target language. Pseudowords that violate them would be too easy to reject in a lexical decision and would not enforce word recognition on real words. We propose a new pseudoword generator, UniPseudo, using an algorithm based on Markov chains of orthographic n-grams. It generates pseudowords from a customizable database, which allows one to control the characteristics of the items. It can produce pseudowords in any language, in orthographic or phonological form. It is possible to generate pseudowords with specific characteristics, such as frequency of letters, bigrams, trigrams or quadrigrams, number of syllables, frequency of biphones, and number of morphemes. Thus, from a list of words composed of verbs, nouns, adjectives, or adverbs, UniPseudo can create pseudowords resembling verbs, nouns, adjectives, or adverbs in any language using an alphabetic or syllabic system.

Pseudowords are stimuli created from letters that look like words but are not words. They differ from nonwords, which are also stimuli created from letters but do not look like words (*CNLME* is an example of a nonword in English). For a long time, pseudowords have been generated by changing one or more letters of an existing word (e.g., *CLEAM* created from *CLEAN* in Davis & Lupker, 2006). Pseudowords are intensively used in psycholinguistics research. They are essential in a task widely used in that field, namely the lexical decision task, where participants have to decide as quickly as possible whether the stimulus presented on the screen is a word. To cause participants to access their mental lexicon, it is essential that the pseudowords respect the phonotactic rules of the language; in other words, that they are potential words.

In creating pseudowords for a lexical decision, it is difficult to avoid pseudowords that are too easy to reject. For example, with nonwords composed only of consonants (such as *DZGLS*), the participants will quickly realize that it is sufficient to detect the presence of a vowel to take the decision and will no longer need to identify the words. In this case reaction times will no longer reflect the processes involved in normal reading. Of course, this example is extreme, but it shows how the construction of pseudowords or nonwords will influence the processing of words. To ensure that pseudowords are well-adapted to the words used in a lexical decision task, it is necessary that they conform to the orthographic rules. Orthographic rules are specific to each language and determine legitimate strings of letters. For example, in English, the letter sequence "*qr*" is not orthographically correct (it does not exist in any English word), whereas "*qu*" is perfectly valid (e.g., *queen*, *quest*, *quarter*).

Another criterion that is relevant when creating pseudowords is the orthographic proximity of the pseudoword with real words. For the English Lexicon Project, Balota et al. (2007) created a lexical decision task with 40,481 words. They created 40,481 matched pseudowords by modifying one or two letters of the original words. This method of creating pseudowords raises the problem of the systematic orthographic proximity of the pseudoword and the original word. Indeed, for some pseudowords, participants can sometimes identify the original word (e.g., *addomen* from *abdomen*). The recognition of the original word can induce an unintended experimental priming bias that is not controlled by the experimenters.

Using the method of replacing one or two letters of the word, the identification of the original word is much easier for long words than for short words, as the percentage of letters common to the pseudoword and the original word is higher in longer words. Short pseudowords have more orthographic neighbors (number of words that can be derived by changing just one letter and preserving letter positions) than long pseudowords. Furthermore, the influence of the original word was notably shown by Yap, Sibley, Balota, Ratcliff, and Rueckl (2015) and Perea, Rosa, and Gómez (2005),

who showed that reaction times to pseudowords are notably influenced by the frequency of the base word.

Being able to recognize the source word of a pseudoword may also affect studies that are specifically interested in the mechanisms used when reading pseudowords. As pseudowords, by definition, should generate limited lexical activity, they can be used to better understand sub-lexical effects. For example, they have been used to study the non-lexical pathway of the dual-route cascaded model (Coltheart, Rastle, Perry, Langdon & Ziegler, 2001), which states that words can be read through a lexical route or a grapheme–phoneme conversion route. In this framework, pseudowords allow one to study the properties of the grapheme–phoneme conversion pathway, because they are necessarily read by the non-lexical pathway (e.g., Proverbio, Vecchi & Zani, 2004). In this situation, it is crucial to generate pseudowords from which the original words are not identifiable. Moreover, before someone learns a word, it is a pseudoword for them. Thus, studying how adults learn pseudowords can reveal the processes involved in word acquisition.

A number of studies have also examined the impact of pseudohomophony (nonwords spelling that can be pronounced like a word) to determine whether there are phonological effects in silent reading (see e.g., Besner & Davelaar, 1983; Taft 1982). Another example of the use of the pseudowords utility is the study from Taft (2004) who manipulated the nature of the pseudowords in a lexical decision to examine whether inflected words are recognized via their stem.

Currently, the main tools used to create pseudowords for research are the ARC Nonword Database, WordGen, and Wuggy. We describe them briefly before presenting our new tool for generating pseudowords.

Existing solutions for the generation of pseudowords

ARC Nonword Database

The ARC Nonword Database, developed by Rastle, Harrington, and Coltheart (2002), contains 358,534 monosyllabic pseudowords, including 48,534 pseudohomophones. To generate these pseudowords, the authors used a grammar based on several phonotactic rules, allowing phonemes to be combined to create potential monomorphemic words. Once these phonological pseudowords were generated, the authors generate the associated orthographic forms using phoneme–grapheme correspondence rules. These pseudowords are grouped in a database that allows the user to select the items according to certain criteria (e.g., the number of neighbors or their bigram frequency [adjacent pairs of letters frequency]). However, the use of this database for research has two important limitations: it is only available in English and contains only monosyllabic pseudowords.

WordGen

WordGen (Duyck, Desmet, Verbeke & Brysbaert, 2004) is a software program that uses the CELEX (Baayen, Piepenbrock, & Gulikers, 1996) and Lexique (New, Pallier, Brysbaert & Ferrand, 2004) databases to generate pseudowords in Dutch, English, German, and French. These pseudowords can be generated according to their number of letters, the size of their neighborhood, or the frequency of their bigrams. To generate a pseudoword, the program concatenates a string of random letters and checks if this string of letters is a word. If it is not a word, it then checks the different properties requested and rejects the string of letters if one of these properties is violated. This means that if the parameters are strict, WordGen may take a long time to find potential candidates. It also means that it may be difficult for researchers not used to word with sub-lexical statistics to find the set of parameters that will allow good pseudowords according to the words in the experiment.

Wuggy

Wuggy (Keuleers & Brysbaert, 2010) is a software program that, in its first incarnation, was able to generate pseudowords in Basque, Dutch, English, French, German, Serbian, and Spanish. The current version of Wuggy (0.3.2) can also generate pseudowords in Italian, Polish, Turkish, and Vietnamese. Wuggy uses a list of syllabified words. These words are decomposed into subsyllabic elements, and these subsyllabic elements are recombined to form pseudowords. To match the created pseudowords with the words given by the psycholinguistic researchers, Wuggy tries to filter the pseudowords with the same frequencies of subsyllabic elements and progressively widens these constraints in terms of frequencies to find a potential candidate. Wuggy is a powerful pseudoword generator, but it is not available for all alphabetic languages, and it produces pseudowords that do not necessarily respect the nature of the words (e.g., generate only present participle pseudowords in English).

None of these three solutions allows one to work from a corpus of words determined entirely by oneself. This ability, however, would allow one to control the properties of the input words and thus determine the properties of the generated pseudowords. For example, from a list of words composed of verbs, nouns, adjectives, or adverbs, one could create pseudowords resembling verbs, nouns, adjectives, or adverbs. This could be particularly interesting for languages in which the grammatical classes have specific orthographic properties. For instance, the present participles of French verbs have a specific ending “ant”. If UniPseudo is given only present participles as input, the generated pseudowords will all end with the specific ending. Whereas with Wuggy this will not be the case because it will recombine sub-lexical elements taken from the whole set of French words and therefore pseudowords without this specific present participle ending will be generated.

Another advantage of generating pseudowords from a list of words provided by the researcher is that it allows one to work with any language, provided that it is alphabetic or syllabic. Current tools are based on orthographic and not phonological algorithms (except Wuggy for English, French, and Italian) and are therefore designed to generate pseudowords for reading or writing research. They cannot easily generate pseudowords for auditory research. A tool using a personalized word list could work directly on phonology whatever the desired language provided that the words in the personalized list would be transcribed in their phonological form (e.g., table -> /teɪbəl/). More recently, two new generators have been proposed using a custom list of words to generate pseudowords, CGCA and UniPseudo, the latter of which is the subject of this article.

CGCA

The CGCA algorithm (König, Calude, & Coxhead, 2020) is the most recent tool. It can take a wordlist or a corpus as input. It extracts all unique tokens from the input to create the origin wordlist. From the origin wordlist, it extracts all possible character-grams (bigram, trigram, 4-gram, 5-gram, etc.) and assigns them to three possible positions: the beginning, middle, or end of words. Finally, it iteratively generates and validates each chain of character-grams. It outputs a list of pseudowords specific to the words used to generate them. CGCA is available as a script on GitHub and needs to be run it in a terminal (which non-programmers are not necessarily familiar with).

UniPseudo

UniPseudo has the following features:

- It is accessible through a web interface (http://unipseudo.lexique.org/).

- The pseudowords it generates are constructed from words given by the user. This property allows the user (1) to generate pseudowords in any language based on an alphabetic or syllabic system, (2) to generate pseudowords with diacritical marks, and (3) to generate pseudowords that respect the orthographic statistics of the input word list. Concerning (2) diacritical letters are considered as different letters and thus UniPseudo will generate pseudowords which probability of appearance of diacritical letters will be similar to the frequency of diacritical letters present in the user's input. Concerning (3), it allows one to generate pseudowords that look like inflected verbs (e.g. from Table 1: beaped), nouns (e.g. from Table 1: encher), or adjectives. It can also generate pseudowords that look like low-frequency words and contain potentially less frequent sublexical parts, or high-frequency words and contain potentially more frequent sublexical parts. Furthermore, it can be used to generate words resembling early- or late-acquired words. Take the example of verb generation in English. If one enters only 5-letter English verbs in the infinitive form, one

obtains as output 5-letter pseudowords that could be verbs in English. An important point is that the proportion of irregular pseudo-verbs in the list should be comparable to that of irregular verbs in the working corpus. Once it has a sufficient list of words of a certain length with certain properties (we recommend at least 100 words, but, in general, the more the better), UniPseudo generates pseudowords that share the properties of these words. Thus, it is possible to generate pseudowords with one or a combination of characteristics (frequency of letters, bigrams, trigrams or quadrigrams, number of syllables, frequency of biphones [adjacent pairs of phonemes], number of morphemes, etc.).

- In addition to generating orthographic pseudowords, the algorithm can also generate phonological pseudowords. Thus, for an auditory lexical decision, it is possible to generate pseudowords with certain phonological characteristics. If phonemes are coded by single Unicode letters, the algorithm will also be able to generate phonological pseudowords. The algorithm operates on single segments sequentially using simple transitional probabilities. For example, to generate pseudowords of 2 syllables starting with an occlusive and ending with a nasal vowel, the user only needs to provide a corpus of input words based on phonological representations rather than orthographic representations of words.
- It also allows words from 64 languages to be easily imported and thus pseudowords to be generated in these 64 languages.
- It can also generate pseudowords in non-Latin alphabets.
- Its source code is available at https://github.com/chrplr/openlexicon/tree/master/apps/unipseudo.

Algorithm used by UniPseudo

UniPseudo uses an algorithm based on Markov chains. It requires a set of words of a given length (e.g., all 6-letter nouns). From this set, it extracts all trigrams (or bigrams if the user decides so) at a given position (e.g., for a word of 6 letters [123456], there are four trigrams: 123, 234, 345, and 456). Then, to create a pseudoword, it starts by randomly selecting a trigram among those starting in initial position (respecting the frequencies in the original corpus). Then it selects randomly a second trigram that start with the two letters finishing the first one, among trigram starting in second position. This process is repeated until the last trigram is reached (the 4th one for a word of 6 letters). Thus, trigrams are selected by the frequency of the input given by the user: a trigram which is frequently in the input will have more probabilities to be selected during the constitution of the pseudowords. This algorithm allows pseudowords to be generated that are similar to the original words. Examples of pseudowords based on 6-letter lemmas that are either nouns or verbs are given in Table 1 in French and English.

Table 1. Examples of 6-letter pseudonouns and pseudoverbs generated with UniPseudo

French		English	
Pseudonouns	**Pseudoverbs**	**Pseudonouns**	**Pseudoverbs**
meston	voirer	encher	beaped
vortin	bouper	gradom	dramed
nivard	garler	canure	geated
nombon	durgir	fortor	suined
bonjon	gicher	gunker	burled
gâtate	redier	ruddle	carned
cognal	palser	dinute	serked
nevoir	flâmer	hercel	fenned
parron	fercir	aspern	hinned
intrit	vanser	twenzy	barted
fignon	dicler	tendom	salked
nectue	dorgir	figlet	yelped
bilice	guiser	borant	hanted
ordeau	bantir	hellot	wooped
ourvie	menter	dinete	wasked
limade	claner	deggon	pusted
taudin	sonter	snique	crayed
partil	tonger	suring	railed
cavure	purler	sullet	tashed
hébris	cester	gaddle	hanked

This algorithm has the advantage that strong constraints can be placed on the word-selection algorithm.

UniPseudo is using an algorithm based on Markov chains similarly to CGCA, although its creation was not accompanied by a specific publication. Indeed, it was already available online in a less modern and less option-rich format on the Lexique website in 2012[1] and it has also already been used to create the pseudowords for the MEGALEX mega-study (Ferrand, Méot, Spinelli, New, Pallier, Bonin, Dufau, Mathôt, & Grainger, 2017). In contrast to CGCA, UniPseudo takes the position of bigrams and trigrams in

[1] http://web.archive.org/web/20120315101935/http://www.lexique.org/toolbox/toolbox.pub/

the word more finely into consideration as far as this is a specific position. CGCA only takes into account the beginning, middle and end positions of the word.

How to use UniPseudo

UniPseudo is available as a web application at the following address: http://unipseudo.lexique.org/. Figure 1 shows the interface.

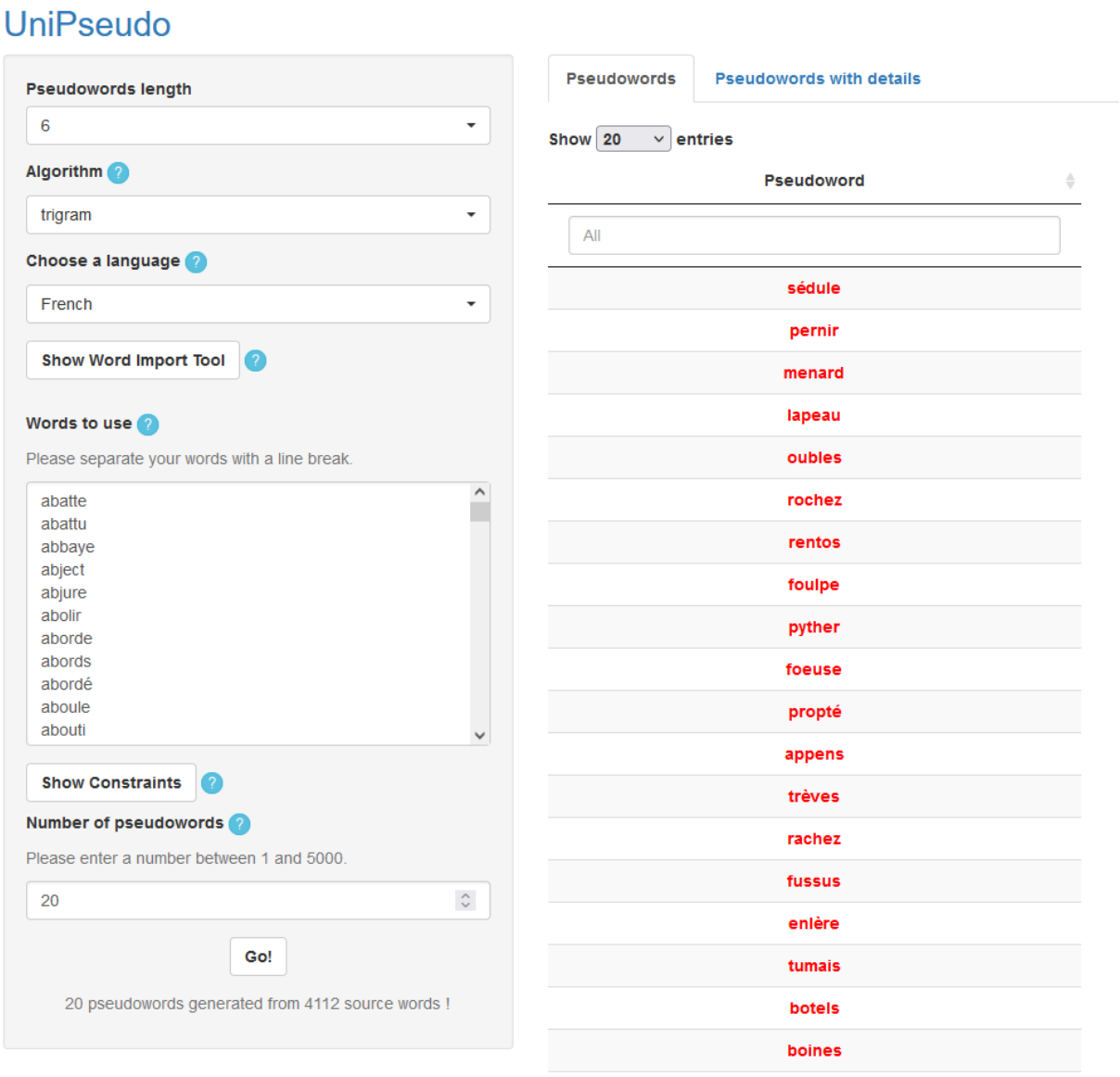


Figure 1: Screen capture of UniPseudo's interface

To use UniPseudo, the user follows these steps:

- Select the desired length of the pseudowords
- Decide whether the algorithm will use trigrams or bigrams.

    - In general, it is advisable to use trigrams. For short words (3, 4, or 5 letters), however, it is preferable to use bigrams, because trigrams yield results that are too similar to existing words.
- Select a language among 64 possibilities.
    - ~~Selecting a language allows the algorithm to apply constraints during the pseudowords generation, namely avoiding words with more than three consecutive consonants or with the same consecutive letter three times or more in Latin languages when using the bigram algorithm.~~
    - Selecting a language also allows the use of the Word Import Tool (see below).

  If these features are not required, the user should choose the first option, "Other."
- Copy the list of words on which the algorithm will base the pseudowords, with a line break separation between each word. To generate word lists, one can use specialized databases for a given language (e.g., Lexique for French and the English Lexicon Database for English). As mentioned earlier, it is recommended to include at least 100 words, but, in general, the more the better. For instance, in Figure 1, we copied all 877 words from Lexique 3.83 that met the following criteria: 6 letters, 2 syllables, lemma (not an inflected form), noun, and a subtitle lemma frequency[2] strictly greater than 1 occurrence per million (so as not to include too rare orthographic forms).
- Choose the number of pseudowords that UniPseudo should generate.
- Use the "Show Constraints" button to set additional constraints to the output if you want to. The user can set constraints such as the maximum number of accented characters possible in a pseudoword, the maximum number of successive consonants or the maximum number of identical letters possible in a pseudoword.
- ~~-~~
- Press the "Go" button.
- You get the requested pseudowords in the UniPseudo window. A manual selection phase of the items is then necessary. If you want for example 50 pseudowords we advise you to generate at least 150 pseudowords to be able to select the 50 best candidates proposed by UniPseudo. If the user performing the manual selection selects on average 1 pseudoword out of 3 as he/she goes through the list, he should minimize possible systematic bias. Indeed, the pseudowords of the list are presented in a completely random order.
- If you want to know the construction steps of each pseudoword, the details are presented step by step in the "Pseudowords details" tab.

[2] The subtitle lemma frequency is, for a word, the sum of the frequency of its inflected and uninflected forms in a subtitle corpus.

When the process is complete, the generated pseudowords appear in the right part of the window. The user can click on “Download Pseudowords” to download an Excel file containing the generated pseudowords.

Once the pseudowords have been generated, the chosen trigram, its position, and the origin word can be displayed by clicking on the “Pseudowords with details” tab (see Appendix 1). For example, the pseudoword *tensus* is composed of the trigram *ten* in first position (from the word *tennis)*, the trigram *ens* in second position (from the word *pensée*), the trigram *nsu* in third position (from the word *consul*), and the trigram *sus* in last position (from the word *dessus*).

Word Import Tool

There is another way to generate pseudowords with UniPseudo without having to first build a custom list of base words. For the 64 languages for which there is a database in WorldLex (Gimenes & New, 2016; see Appendix 2 for the list of the 64 languages), it is possible to automatically import a word list of the desired length into UniPseudo (see Figure 2). Only words with a frequency strictly greater than 0.5 are imported so as not to include too rare orthographic forms.

UniPseudo

Pseudowords length

8

Algorithm

trigram

Choose a language

English

Hide Word Import Tool

Use words from database

Words to use

Please separate your words with a line break.

actually
children
together
anything
everyone
probably
remember
thinking
favorite
although

Show Constraints

Number of pseudowords

Please enter a number between 1 and 5000.

20

Go!

Figure 2. Screen capture of the Word Import Tool

To use the Word Import Tool, the user follows these steps:

- Choose the length of the pseudowords.
- Choose whether the algorithm will use trigrams or bigrams.
- Choose a language among 64 possibilities.
- Click on “Show Word Import Tool.”
- Click on “Use words from database” to import words of the selected language and chosen length.
- Choose the number of pseudowords desired.
- Choose if you want to use custom constraints.
- Click on “Go!”

Comparison of UniPseudo with existing tools

First, we compared the quality of the pseudowords proposed by UniPseudo to those proposed by Wuggy and CGCA. For this, we used the LD1NN algorithm (Keuleers & Brysbaert, 2011). This algorithm assesses how easy (or not) it is for a participant to determine whether the presented stimulus is a pseudoword. For each stimulus in a lexical decision, LD1NN calculates the Levenshtein distance (the minimum number of operations, i.e. deletions, insertions or substitutions required to turn one word into the other) between the current stimulus and the previous ones. This allows it to identify the stimuli closest to the current stimulus. Finally, it calculates the probability of giving a word response based on the relative frequency of words and pseudowords in this close neighborhood.

To test the relevance of the pseudowords generated by UniPseudo, we selected from Lexique 3.83 all the French words of 8 letters having a lemma frequency higher than 1 and containing neither space, dash, nor apostrophe. From these words, we generated 2,400 pseudowords with CGCA and UniPseudo using trigrams. Then we randomly selected 2,400 words and used Wuggy 0.2.2b2 to generate only one pseudoword per selected word using the default options (“match length of subsyllabic segments,” “match letter length,” “match transition frequencies,” and “match subsyllabic segments 2 out of 3”). The word and pseudoword lists used are available on OSF.[3] Once the three lists were generated (same 2,400 words and 2,400 different pseudowords for the three tools), we randomized their order and then applied the LD1NN algorithm to the resulting word and pseudoword list. The results are presented in Figure 3.

[3] https://osf.io/svtq5/?view_only=08894231bc094ba7853f335373056a10

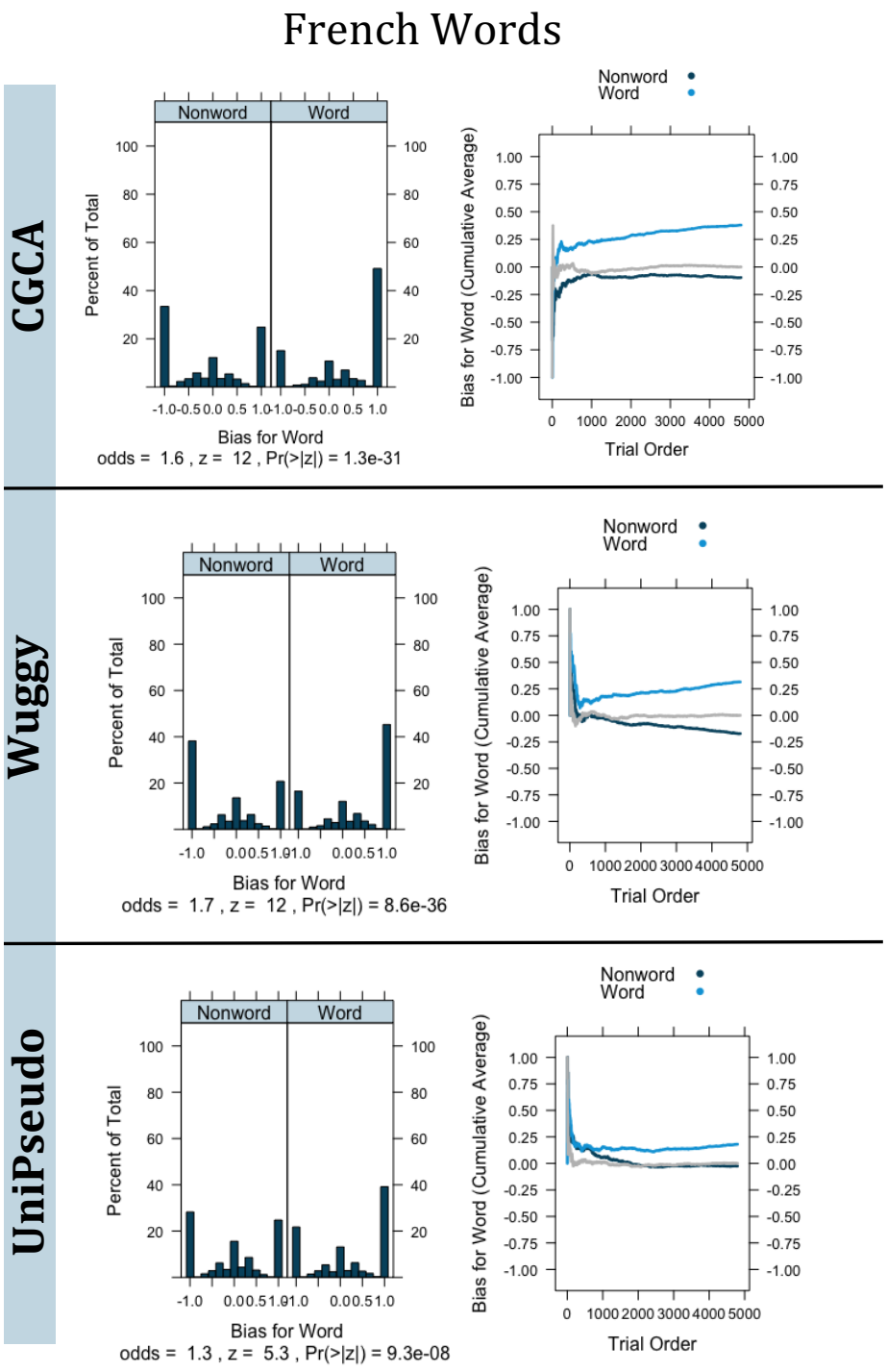


Figure 3. Results of LD1NN algorithm presented with a group of 2,400 French 8-letter words and 2,400 8-letter nonwords created using CGCA, Wuggy, and UniPseudo algorithms. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the gray line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

On the left panel of Figure 3 each vertical bar should be read as the percentage of nonword stimuli (left subpanel) and word stimuli (right subpanel) that had a particular bias for a word response (positive) or for a nonword response (negative). As shown on the left panel of Figure 3, the probability of a nonword being considered a nonword (the leftmost bar on the left subpanel) is less strong for UniPseudo than for CGCA or Wuggy. A pseudoword is a good stimulus if it looks like a word. Indeed the answer to decipher whether a stimulus is a word should require lexical access and not only the detection of surface features of the pseudoword construction.

In the right panel, one can see that LD1NN increasingly detects words more effectively in the lists generated by CGCA. When the pseudowords are made by Wuggy, the algorithm also tends to detect words and pseudowords increasingly effectively, but the bias remains lower than that of CGCA, especially for pseudowords. For UniPseudo, one can observe a reduced bias for both words and pseudowords, and this bias does not tend to increase with each trial.

We performed another analysis similar to the one described above but with lemmas (uninflected forms) only. We selected all 8-letter words that were lemmas without spaces, hyphens, or apostrophes and had a subtitle lemma frequency of at least 1 occurrence per million words. We used these words to create 1,800 pseudowords with CGCA and UniPseudo. Then we randomly selected 1,800 words from which we created 1,800 pseudowords with Wuggy by asking it to generate only one candidate with the default options. Once the 1,800 words and 1,800 pseudowords were generated, we randomized their order and then applied the LD1NN algorithm to the list of words and pseudowords obtained. Again, the results on the right-hand side of Appendix 3 show that LD1NN is better at guessing that the pseudowords are pseudowords when they are created by CGCA and Wuggy than when they are generated by UniPseudo. Moreover, as the trials progress, LD1NN is better at discriminating between words and pseudowords when the pseudowords come from Wuggy and CGCA than when they come from UniPseudo. The logistic regression indicates that LD1NN distinguishes words from pseudowords when the pseudowords are made with CGCA ($z = 6.7$, $p < .001$) or Wuggy ($z = 9.4$, $p < .001$), whereas it does not distinguish words from non-words when the pseudowords are made with UniPseudo ($z = 0.1$, $p = .92$).

Finally, we ran simulations in English to check that these results were not due to some peculiarities of French. We selected English words from WorldLex[4] with a surface frequency of at least 1 occurrence per million words. The results are presented in Appendices 4 (all words) and 5 (English lemmas). The results are similar to the results observed in French. LD1NN has more difficulty discriminating between words and pseudowords when the pseudowords are made by UniPseudo than when they are made by Wuggy or CGCA, although the performances of UniPseudo and Wuggy are

[4] http://worldlex.lexique.org

more similar than in previous simulations. In general, LD1NN has more difficulty discriminating words from pseudowords with the English lexical decisions than with the French ones, suggesting that it is easier to generate good pseudowords in English than in French. Compared with Wuggy and CGCA, UniPseudo produces particularly interesting results, because it combines n-grams whose position in the original word list is preserved. This allows it to favor pseudowords with a valid structure in all languages without the need for additional checks.

Supplementary analyses (Appendix 6, 7, 8, 9) were conducted on the pseudowords generated by CGCA, Wuggy and UniPseudo in order to investigate the quality of the different pseudowords lists. For the word and pseudoword lists that were generated for the previous 4 analyses (LD1NN), we calculated frequency indices for the words and for the pseudowords. For French words and lemmas, we used Lexique-Infra (Gimenes, et al., 2020) to compute the frequency of letter types and tokens, bigram types and tokens and trigram types and tokens. For English words and lemmas we used N-Watch (Davis, 2005) to compute the frequency of bigram types and tokens and the frequency of trigram types and tokens. The type frequency corresponds to the number of occurrences of a given string (letter, bigram or trigram) in a corpus, while token frequency weighted the number of occurrences by the frequency of the words. The properties of the words and pseudowords were then compared using Welch's t-test because the homogeneity of variances were not respected (Brow-Forsythe test: $p < .05$). A significant test indicated statistically different properties between the word list and the pseudoword list. Overall, the analyses (Appendix 6, 7) show similar results to those obtained with LD1NN confirming the quality of the pseudowords generated by UniPseudo.

In line with König et al. (2020), two different researchers manually coded the number of polymorphic, near polymorphic and one-character dissimilarity for the first 100 8-letter pseudowords generated by CGCA, Wuggy or UniPseudo in our previous French words analysis. König et al. (2020) defined polymorphic pseudowords as pseudowords that consist of a real root plus one of more affixes, near polymorphic pseudowords as pseudowords whose root does not exist in the language but include at least one affix, and one-character dissimilarity pseudowords as pseudowords that are one character away from at least one real word within the language[5]. In order to be able to compare the index "one-character dissimilarity" was also computed for the first 100 words used in our previous French words analysis. The results are presented in Table 2.

Table 2. Results from the comparison suitability evaluation (per 100 pseudowords)

	Polymorphic	Near polymorphic	Char Dissimilarity
UniPseudo	19	51	32
CGCA	11	52	13
Wuggy	7	52	6
Words	-	-	70

[5] Lexique-Infra (Gimenes, et al., 2020) was used to compute one-character dissimilarity

The polymorphic pseudowords generated by UniPseudo, have higher counts (19%) than any of the other pseudowords generators (11% for CGCA and 7% for Wuggy) suggesting that UniPseudo pseudowords are more word-like. For the near polymorphic pseudowords counts do not differ across the different generators. Finally, the number of pseudowords being one character away from a real word is higher for UniPseudo (32%) than for CGCA (13%) and Wuggy (6%). Wuggy results in a lower number than CGCA. Interestingly, words tend to have more neighbours (70%) than all generators.

Conclusion

UniPseudo is an algorithm for generating pseudowords from a customizable database that allows the user to closely control the characteristics of the input items. It produces pseudowords in any language based on orthographic or phonological forms or on any other string representation of the input word base. Compared with the existing tools, UniPseudo generated pseudowords that were closer (according to the LD1NN algorithm) to the source words. This powerful new pseudoword generation tool is freely available and should be a valuable tool to facilitate the work of psycholinguists.

References

Baayen, R. H., Piepenbrock, R., & Gulikers, L. (1996). The CELEX lexical database (cd-rom).

Balota, D. A., Yap, M. J., Hutchison, K. A., Cortese, M. J., Kessler, B., Loftis, B., ... & Treiman, R. (2007). The English lexicon project. *Behavior research methods, 39(3)*, 445-459.

Besner, D., & Davelaar, E. (1983). Suedohomofoan effects in visual word recognition: evidence for phonological processing. *Canadian Journal of Psychology/Revue canadienne de psychologie*, *37*(2), 300.

Brysbaert, M. (2010). Wuggy: A multilingual pseudoword generator. *Behavior Research Methods 42(3)*, 627-633.

Coltheart, M., Rastle, K., Perry, C., Langdon, R., & Ziegler, J. (2001). DRC: a dual route cascaded model of visual word recognition and reading aloud. *Psychological review, 108(1)*, 204.

Davis, C. J. (2005). N-Watch: A program for deriving neighborhood size and other psycholinguistic statistics. *Behavior research methods*, *37*(1), 65-70.

Davis, C. J., & Lupker, S. J. (2006). Masked inhibitory priming in English: Evidence for lexical inhibition. *Journal of Experimental Psychology: Human Perception and Performance, 32(3)*, 668.

Duyck, W., Desmet, T., Verbeke, L. P., & Brysbaert, M. (2004). WordGen: A tool for word selection and nonword generation in Dutch, English, German, and French. *Behavior Research Methods, Instruments, & Computers, 36(3)*, 488-499.

Ferrand, L., Méot, A., Spinelli, E., New, B., Pallier, C., Bonin, P., ... & Gimenes, M., & New, B. (2016). Worldlex: Twitter and blog word frequencies for 66 languages. *Behavior research methods, 48(3),* 963-972.

Gimenes, M., Perret, C., & New, B. (2020). Lexique-Infra: grapheme-phoneme, phoneme-grapheme regularity, consistency, and other sublexical statistics for 137,717 polysyllabic French words. *Behavior Research Methods*, *52*(6), 2480-2488.

Grainger, J. (2018). MEGALEX: A megastudy of visual and auditory word recognition. *Behavior Research Methods, 50(3)*, 1285-1307.

Keuleers, E., & Brysbaert, M. (2010). Wuggy: A multilingual pseudoword generator. *Behavior research methods, 42(3)*, 627-633.

König, J., Calude, A. S., & Coxhead, A. (2020). Using character-grams to automatically generate pseudowords and how to evaluate them. *Applied Linguistics, 41(6)*, 878-900.

New, B., Pallier, C., Brysbaert, M., & Ferrand, L. (2004). Lexique 2: A new French lexical database. *Behavior Research Methods, Instruments, & Computers, 36(3)*, 516-524.

Perea, M., Rosa, E., & Gómez, C. (2005). The frequency effect for pseudowords in the lexical decision task. *Perception & Psychophysics, 67(2)*, 301-314.

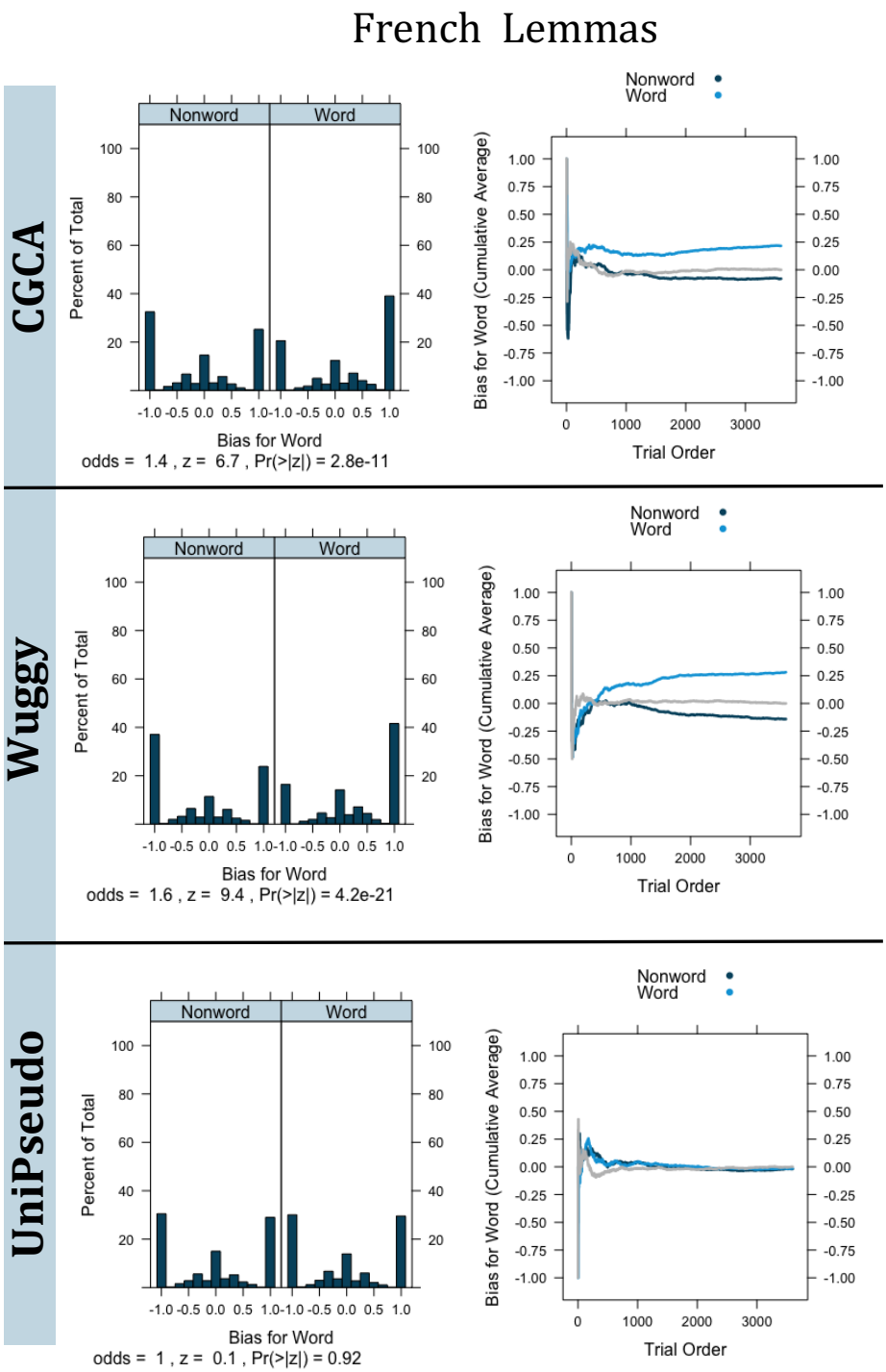
Proverbio, A. M., Vecchi, L., & Zani, A. (2004). From orthography to phonetics: ERP measures of grapheme-to-phoneme conversion mechanisms in reading. *Journal of cognitive neuroscience, 16(2)*, 301-317.

Taft, M. (1982). An alternative to grapheme-phoneme conversion rules?. *Memory & Cognition*, *10*(5), 465-474.
Taft, M. (2004). Morphological decomposition and the reverse base frequency effect. *The Quarterly Journal of Experimental Psychology Section A*, *57*(4), 745-765.
Yap, M. J., Sibley, D. E., Balota, D. A., Ratcliff, R., & Rueckl, J. (2015). Responding to nonwords in the lexical decision task: Insights from the English Lexicon Project. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 41(3)*, 597.

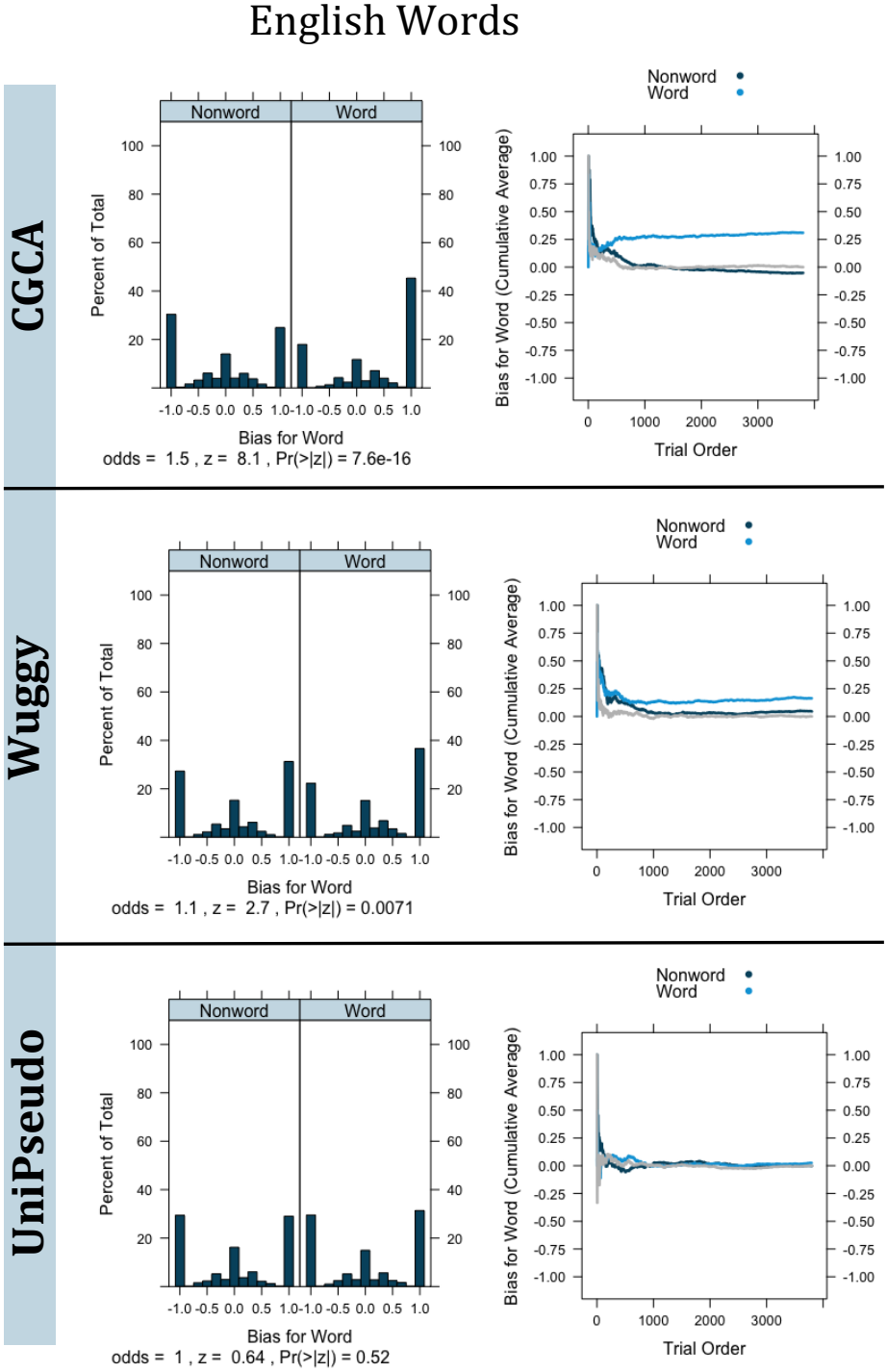
Appendix 1. Screen capture of the "Pseudowords with details" tab

Afrikaans	Catalan	French	Hungarian	Latvian	Persian	Sinhala	Tamil
Albanian	Croatian	Georgian	Icelandic	Lithuanian	Polish	Slovak	Telugu
Amharic	Czech	German	Indonesian	Macedonian	Portuguese Brazil	Slovenian	Turkish
Arabic	Danish	Greek	Italian	Malayalam	Portuguese Europe	Spanish Latin America	Ukrainian
Armenian	Dutch	Greenlandic	Japanese	Malaysian	Punjabi	Spanish Spain	Urdu
Azeri	English	Gujarati	Kannada	Mongolian	Romanian	Swahili	Uzbek
Bengali	Estonian	Hebrew	Kazakh	Nepali	Russian	Swedish	Vietnamese
Bosnian	Finnish	Hindi	Khmer	Norwegian	Serbian	Tagalog	Welsh

Appendix 2. List of the 64 languages for which words can be automatically selected in UniPseudo

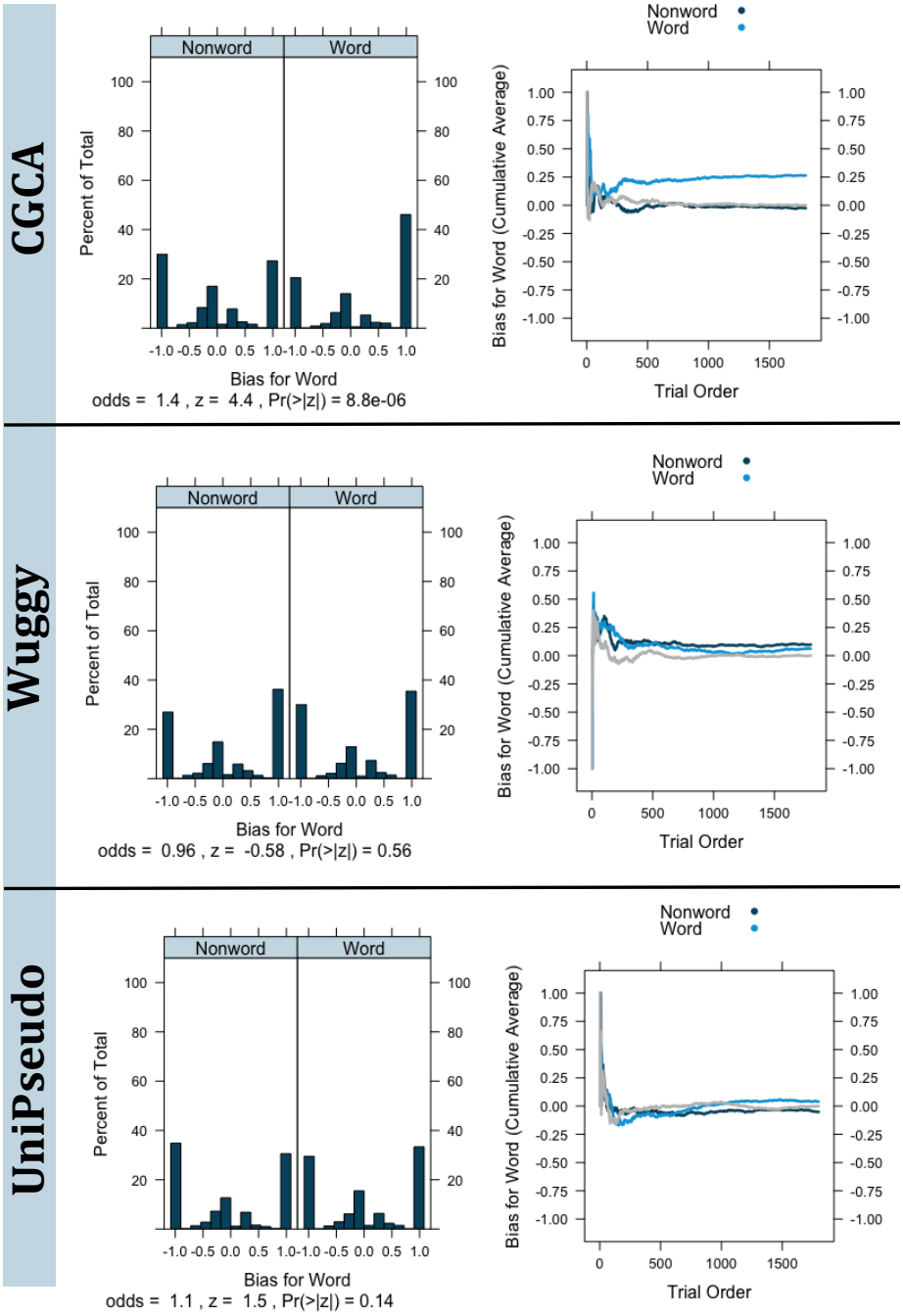


Appendix 3. Results of LD1NN algorithm presented with a group of 1800 French 8-letter lemma words and 1800 8-letter nonwords created from CGCA, Wuggy and UniPseudo algorithms. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.



Appendix 4. Results of LD1NN algorithm presented with a group of 1900 English 8-letter words and 1900 8-letter nonwords created from CGCA, Wuggy and UniPseudo algorithms. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

English Lemmas



Appendix 5. Results of LD1NN algorithm presented with a group of 900 English 8-letter lemma words and 900 8-letter nonwords created from CGCA, Wuggy and UniPseudo algorithms. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

	French Words				French Lemmas			
		t	df	p		t	df	p
CGCA	Letter Type	-5.582	4797.326	< .001	Letter Type	-0.639	3574.176	0.523
	Letter Token	-6.545	4792.186	< .001	Letter Token	-0.611	3597.02	0.541
	Bigram Type	-9.634	4794.32	< .001	Bigram Type	-4.068	3597.973	< .001
	Bigram Token	-8.779	4750.599	< .001	Bigram Token	-5.112	3571.483	< .001
	Trigram Type	-7.214	4749.731	< .001	Trigram Type	-2.974	3567.116	0.003
	Trigram Token	-7.847	4748.448	< .001	Trigram Token	-5.376	3496.002	< .001
Wuggy	Letter Type	-0.301	4787.408	0.764	Letter Type	-1.66	3597.537	0.097
	Letter Token	-0.729	4791.874	0.466	Letter Token	-1.058	3597.989	0.29
	Bigram Type	-6.918	4793.401	< .001	Bigram Type	-7.103	3597.692	< .001
	Bigram Token	-4.765	4790.661	< .001	Bigram Token	-6.452	3569.575	< .001
	Trigram Type	-8.96	4792.782	< .001	Trigram Type	-9.835	3573.719	< .001
	Trigram Token	-6.209	4795.883	< .001	Trigram Token	-9.997	3516.721	< .001
UniPseudo	Letter Type	1.058	4797.463	0.29	Letter Type	-0.983	3587.276	0.326
	Letter Token	0.235	4794.207	0.814	Letter Token	-1.001	3597.845	0.317
	Bigram Type	-0.722	4797.988	0.47	Bigram Type	-0.31	3595.801	0.756
	Bigram Token	-0.567	4792.167	0.571	Bigram Token	0.346	3597.669	0.73
	Trigram Type	-1.83	4775.106	0.067	Trigram Type	-0.963	3593.159	0.336
	Trigram Token	-0.729	4797.876	0.466	Trigram Token	-0.607	3592.913	0.544

Appendix 6. Left panel: Results of Welch t-tests on the letter, bigram and trigram statistics computed on 2400 French 8-letter words and 2400 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms. Right panel: Results of Welch t-tests on the letter, bigram and trigram statistics computed on 1800 French 8-letter lemma words and 1800 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms.

		French Words		French Lemmas	
		Means	**SD**	**Means**	**SD**
Words	Letter Type	52132	8161	50301	7206
	Letter Token	122499	19562	120165	18109
	Bigram Type	5884	2141	4958	1780
	Bigram Token	11677	5170	10398	4317
	Trigram Type	974	721	718	540
	Trigram Token	1551	1255	1366	1020
CGCA Pseudowords	Letter Type	50833	8017	50140	7821
	Letter Token	118927	18753	119793	18410
	Bigram Type	5301	2070	4717	1785
	Bigram Token	10451	4644	9692	3960
	Trigram Type	832	646	667	492
	Trigram Token	1285	1124	1197	858
Wuggy Pseudowords	Letter Type	52066	7813	49901	7277
	Letter Token	122108	18979	119527	18046
	Bigram Type	5460	2090	4539	1761
	Bigram Token	10978	4994	9508	3941
	Trigram Type	789	702	548	497
	Trigram Token	1327	1247	1049	873
UniPseudo Pseudowords	Letter Type	52333	7962	50058	7612
	Letter Token	122560	18758	119558	18228
	Bigram Type	5844	2116	4940	1825
	Bigram Token	11599	4918	10448	4276
	Trigram Type	943	666	701	521
	Trigram Token	1533	1233	1346	982

Appendix 7. Left panel: Means and SD for the letter, bigram and trigram statistics computed on 2400 French 8-letter words and 2400 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms. Right panel: Means and SD for the letter, bigram and trigram statistics computed on 1800 French 8-letter lemma words and 1800 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms.

	English Words				English Lemmas			
		t	df	p		t	df	p
CGCA	Bigram Type	-8.691	3741.083	< .001	Bigram Type	-1.647	1792.669	0.1
	Bigram Token	-7.845	3719.334	< .001	Bigram Token	-1.821	1787.212	0.069
	Trigram Type	-10.343	3616.261	< .001	Trigram Type	-4.391	1796.926	< .001
	Trigram Token	-8.382	3604.178	< .001	Trigram Token	-5.192	1795.057	< .001
Wuggy	Bigram Type	-3.763	3796.451	< .001	Bigram Type	-4.056	1797.347	< .001
	Bigram Token	-2.285	3797.88	0.022	Bigram Token	-2.094	1789.823	0.036
	Trigram Type	-3.784	3797.906	< .001	Trigram Type	-8.906	1736.027	< .001
	Trigram Token	-1.898	3797.9	0.058	Trigram Token	-8.947	1798	< .001
UniPseudo	Bigram Type	-1.171	3796.118	0.242	Bigram Type	3.569	1790.452	< .001
	Bigram Token	-0.912	3796.966	0.362	Bigram Token	3.189	1784.594	0.001
	Trigram Type	-0.945	3796.63	0.345	Trigram Type	3.37	1746.204	< .001
	Trigram Token	-0.712	3796.51	0.476	Trigram Token	3.277	1771.565	0.001

Appendix 8. Left panel: Results of Welch t-tests on the letter, bigram and trigram statistics computed on 1900 English 8-letter words and 1900 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms. Right panel: Results of Welch t-tests on the letter, bigram and trigram statistics computed on 900 French 8-letter lemma words and 900 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms.

		English Words		English Lemmas	
		Means	**SD**	**Means**	**SD**
Words	Bigram Type	870.5	586.1	462.1	238.8
	Bigram Token	116.0	86.2	53.1	24.3
	Trigram Type	285.1	327.0	101.1	96.5
	Trigram Token	38.1	50.8	9.6	6.5
CGCA Pseudowords	Bigram Type	714.5	517.8	462.1	226.1
	Bigram Token	95.5	74.5	53.1	22.5
	Trigram Type	186.0	260.3	101.1	94.2
	Trigram Token	25.7	40.1	9.6	6.2
Wuggy Pseudowords	Bigram Type	799.6	574.4	464.4	256.9
	Bigram Token	109.7	85.8	55.3	27.8
	Trigram Type	244.9	328.6	91.4	97.3
	Trigram Token	35.0	51.1	9.4	6.9
UniPseudo Pseudowords	Bigram Type	848.5	573.2	521.7	254.8
	Bigram Token	113.5	84.8	58.9	26.5
	Trigram Type	275.2	320.8	137.7	114.9
	Trigram Token	36.9	49.8	12.3	7.3

Appendix 9. Means and SD for the letter, bigram and trigram statistics computed on 1900 English 8-letter words and 1900 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms. Right panel: Means and SD for the letter, bigram and trigram statistics computed on 900 French 8-letter lemma words and 900 8-letter pseudowords created from CGCA, Wuggy and UniPseudo algorithms.