Detecting inherent bias in lexical decision experiments with the LD1NN algorithm

Emmanuel Keuleers and Marc Brysbaert

Ghent University, Belgium

Address for correspondence:

Emmanuel Keuleers

Ghent University, Department of Experimental Psychology

Henri Dunantlaan 2

B-9000 Gent

Belgium

emmanuel.keuleers@ugent.be

Preprint of chapter published as: Keuleers, E., & Brysbaert, M. (2012). Detecting inherent bias in lexical decision experiments with the LD1NN algorithm. In G. Libben, G. Jarema, & C. Westbury (Eds.), *Methodological and Analytic Frontiers in Lexical Research* (pp. 231–248). John Benjamins Publishing.

Abstract

A basic assumption of the lexical decision task is that a correct response to a word requires access to a corresponding mental representation of that word. However, systematic patterns of similarities and differences between words and nonwords can lead to an inherent bias for a particular response to a given stimulus. In this paper we introduce LD1NN, a simple algorithm based on one-nearest-neighbor classification that predicts the probability of a word response for each stimulus in an experiment by looking at the word/nonword probabilities of the most similar previously presented stimuli. Then, we apply LD1NN to the task of detecting differences between a set of words and different sets of matched nonwords. Finally, we show that the LD1NN word response probabilities are predictive of response times in three large lexical decision studies and that predicted biases for and against word responses corresponds with respectively faster and slower responses to words in the three studies.

The lexical decision task in psycholinguistics

The lexical decision task is one of the most popular tasks in psycholinguistics. Participants in visual lexical decision experiments must decide as quickly as possible whether a letter string presented on a computer screen is a word or not by pressing a button corresponding to each of those answers. If the presented letter string is a word, the time between the presentation of the stimulus and the time the button press is initiated, is taken as a measure of the accessibility of that word in the mental lexicon.

The popularity of the visual lexical decision task is undoubtedly because it is a very cheap method, from its implementation to the analysis of its results. As the stimuli are letter strings, the experimental materials, once chosen, require no further effort to produce, in contrast to tasks where, for instance, recorded words or pictures are used. The required experimental setup is also very basic, consisting of a computer and a device to collect key presses. The experiment is easy to explain to participants and many responses can be collected in a single experimental session. For instance, in a large scale lexical decision study where responses were collected for most mono- and disyllabic word in Dutch, Keuleers, Diependaele, and Brysbaert (2010) found that most participants were able to process nearly 2000 stimuli in a one hour session. Finally, the collected data —reaction times (RTs) and accuracies— are easily interpretable without further processing, and are analyzable using well-known and powerful statistical methods.

Despite all these advantages, the lexical decision task presents the experimenter with a number of methodological challenges. For instance, if the task is used to examine differences in processing time between different groups of words (for instance animate vs. inanimate nouns, verbs vs. adjectives, etc.), then the words in both groups must be matched on several other variables, such as word frequency, word length, number of syllables, neighborhood density, and other variables well known to affect lexical decision RTs.[1] In this paper we will focus on an even more basic but rather overlooked problem, namely that, in order to be valid, a lexical decision task requires that the words and nonwords should not be discriminable without knowledge of the words in that language. Imagine a participant who is doing a lexical decision experiment in a language that is unknown to him. If after a while, that participant, given feedback on the lexicality of previously presented stimuli, starts to respond above chance level, he is clearly not doing a task that requires lexical access. Instead, he learns to do stimulus discrimination task based on inherent systematic differences between the two types of stimuli.

Discriminating between words and nonwords without access to the mental lexicon

The lexical decision task hinges on the assumption that in order to decide that a stimulus is a word participants must access a mental representation of that word. For this assumption to hold, participants in a lexical decision experiment must not be

[1] Word frequency, in particular, has such a large influence on lexical decision RTs that researchers presenting results from lexical decision experiments are routinely asked the devastating question '*Are you sure it's not a frequency effect?*' referring to the fact that a difference between two conditions very often turns out to be due to a difference in word frequencies instead of due to a difference in the variable the researchers think they are studying.

able to discriminate word stimuli from nonword stimuli solely on the merits of differences and/or similarities between and within the two types of stimuli. If they are able to do so, the lexical decision task can possibly be performed without lexical access. The reaction times to words would no longer reflect the time course of lexical access, rendering the results unreliable.

Let us imagine a computer algorithm *without any previous knowledge of words* that is presented with the stimuli from a lexical decision task in the order in which they are presented to the participant. The algorithm's task is to decide whether each presented stimulus is a word or not, keeping only a record of the presented stimuli and whether each one was a word or a nonword. If the algorithm predicts above chance level which stimuli were words and which were not, we know that there exists a method that does not require knowledge of words in the language to discriminate between the words and nonwords in the experiment.

In this paper, we present a very simple algorithm that performs the task of discriminating between words and nonwords in a lexical decision experiment. At the core of this algorithm are two widely known machine-learning methods. The first is *one nearest neighbor classification* (1-NN), which tries to predict the class of a novel stimulus by looking at the majority class of its nearest neighbors, given some metric defining the distance between two stimuli (Fix & Hodges, 1951). The second is the *Levenshtein distance* (Levenshtein, 1966), which defines the distance between two strings as the minimal number of substitutions, deletions, and insertions of characters required to transform one string in the other. For instance, the Levenshtein distance between the strings "alcirans" and "walfine" is 5, because a

minimum number of 5 steps are required to transform one into the other: (1) replacing the final character of 'alcirans' by 'e' gives 'alcirane', (2) deleting the 5th character gives 'alcirne', (3) again deleting the 5th character gives 'alcine', (4) replacing the third character gives 'alfine', and (5) inserting 'w' before the 1st character gives 'walfine'.

Combining *one nearest neighbor classification* with the Levenshtein distance produces an algorithm, which we will call *LD1NN,* and which can be described as follows:

1. Compute the Levenshtein distances between the currently presented stimulus and all previously presented stimuli.
2. Identify the previously presented stimuli that are at the nearest distance from the current stimulus.
3. Compute the probability of a *word* response for the given stimulus based on the relative frequency of words among the nearest neighbors.

Table 1 illustrates the application of the ED1NN algorithm to a set of 10 randomly chosen stimuli.

Insert Table 1 here

In the remainder of this paper, we first apply the LD1NN algorithm to mock datasets, examining the extent to which it can successfully discriminate between words and different kinds of nonwords. Following that, we apply LD1NN to data from three large-scale lexical decision experiments, showing that there is a significant inherent bias for or against giving a yes response to word stimuli in all these experiments.

Finally, we establish that the classification probabilities derived from LD1NN are also significant predictors of RTs in the mentioned experiments.

Applying the LD1NN algorithm to mock data

To test the usefulness of the LD1NN algorithm we first applied it to some sets of mock stimuli.

Words vs. random letter strings

The first set of stimuli we tested consisted of a set of 2500 English word stimuli with lengths of 4 to 10 letters, selected at random from the British Lexicon project (Keuleers, Lacey, Rastle, & Brysbaert, under revision) and 2500 random letter strings, matching the words on distribution of lengths (i.e., as many nonwords of length 4 as words of length 4, etc.). Nonwords were constructed by first choosing a length between 4 and 10, and then giving equal probability to each of the 26 letters of the alphabet at each character position, resulting in stimuli such as *bqed, tvhdfeav, wxzeacmquz, bilxb,* or *mfofumfsz*. The word and nonword stimuli were presented to the LD1NN algorithm in randomized order.

Figure 1 shows that LD1NN easily detected the systematic difference between words and the random letter strings. Each vertical bar in the histograms should be read as the percentage of word stimuli (left subpanel) and nonword stimuli (right subpanel) that had a particular bias for a word response (positive) or for a nonword response (negative). The left subpanel shows that more than 80 % of all word stimuli in the examined set had a full bias for the word response (all the nearest neighbors ED1NN found for these stimuli were words). In contrast, the right subpanel shows that only

15 % of all nonwords had a complete bias for a word response, while almost 40 % of nonwords had a full bias against a word response (all the nearest neighbors ED1NN found for these stimuli were nonwords). It is clear that the distributions of bias for word responses are so different for words and nonwords that a correct prediction can be made in most cases. The right panel of Figure 1 illustrates this. At the beginning of the experiment, the numerical bias (the grey line), is slightly in favor of words, simply meaning that the proportion of word stimuli was higher at the beginning of the experiment. However, the average word response bias for word stimuli rises to nearly 90%, meaning that it is possible for LD1NN to identify words with 90% accuracy. It is harder for the algorithm to correctly reject nonwords, but the bias is still very substantial (over 25% better than chance). As expected, a logistic regression indicates that word bias is a significant predictor of stimulus type (z = 26, p < .001). The direction coefficient for the word response bias can also be used to calculate the odds for a word decision: A stimulus for which LD1NN predicts a word response, is 2.9 times more likely to be a word than a stimulus for which LD1NN predicts a nonword response.

Insert Figure 1 here

Discriminating between words and ARC nonwords

The example above indicates that it is easy to distinguish word from nonwords if the nonwords are random letter strings. For empirical evidence that these nonwords lead to fast responses and small frequency effects in psychological experiments, see

Gibbs and Van Orden (1998) or Ghyselinck, Lewis, and Brysbaert (2004), among many others. A lot of effort has been put in making nonwords that should make lexical decision tasks better. One of the most systematic efforts is the ARC nonword database (Rastle, Harrington, & Coltheart, 2001), a collection of 358,534 English pseudowords (nonwords that are phonologically and orthographically legal in English). Typical examples of nonwords in the ARC database are *twund, blerm, prause, shroaze, and splitch.*

We ran the LD1NN algorithm on the words used in the previous simulation, but now paired them with nonwords taken from the ARC online database (http://www.maccs.mq.edu.au/~nwdb/), selecting only nonwords with existing onsets and legal bigrams and making sure that the distribution of word lengths matched the distribution of nonword lengths.

Figure 2 shows the results of the simulation, which are quite surprising. While the algorithm does not detect words as well as in the experiment with random letter strings, it is much easier to distinguish nonwords from words than in the previous simulation. It is 4.1 more likely that a stimulus is a word when LD1NN predicts a word response than when the algorithm predicted a nonword response. A likely explanation for this result is that the ARC nonwords are more similar to each other than the random letter strings are to each other, so that LD1NN is more likely to find nonword neighbors for the ARC nonwords than for the random letter strings, which are more heterogeneously distributed in letter space. Therefore, even if some words are more similar to ARC nonwords than to random letter strings, the net result is that discrimination is easier when pairing words with ARC nonwords. It should be

noted that the ARC database provides a whole range of options for selecting nonword stimuli based on specific criteria that, when carefully used, can probably result in better nonword stimuli. We only took random samples matched for length.

Insert Figure 2 here

Discriminating between words and Wuggy nonwords

The simulation above showed that the LD1NN algorithm easily detected the difference between words and randomly selected nonwords from the ARC nonword database. While there is no immediate evidence that participants in a lexical decision experiment would also find it so easy to detect this difference, in principle they can and so it is a wise precaution to use additional constraints when making nonwords for an LD experiment. One of the methods to make nonwords more alike to words is by making sure bigrams in the words statistically match bigrams in the nonwords. This is a very laborious process if done manually. Another simple and common method is to change one or two letters in existing words. This should give very good results, as nonwords are guaranteed to be very similar to existing words. However, as we will see later, this may lead to other problems, especially when the nonwords are used in the same experiment as the words they are based on.

In Keuleers and Brysbaert (2010), we proposed a method to generate nonwords for lexical decision experiments in a more principled way.[2] The Wuggy algorithm makes

[2] The Wuggy algorithm was created out of necessity. As we were setting up lexical decision experiments with more than 28,000 trials per participant (Keuleers,

the task of generating nonwords for an experiment with given nonwords much easier. Wuggy's default options are to generate nonwords that match input words on length, on subsyllabic structure, and on transition frequencies between subsyllabic segments. In addition, the nonwords have a required 2/3 subsyllabic overlap with the words they are based on, so that these nonwords cannot be easily recognized as being based on particular words (as is the case when one or two letters are changed in longer words).

We applied the LD1NN algorithm to a mock experiment containing the 2500 words used earlier and 2500 matching nonwords generated by Wuggy. Out of the 10 nonwords proposed by Wuggy for each word, we automatically chose its best proposed match, unless the word was an inflected form, in which case we gave preference to a nonword matching the inflectional suffix (such as the *–s* in houses, or the *–ed* in walked).

As can be seen from Figure 3, the word bias is sharply reduced compared to the previous simulations. Both distributions are very similar, and the odds are also much closer to 1, indicating that the stimuli are far more balanced. However, there is still some bias present and the algorithm is able to discriminate words from nonwords better than chance. Interestingly, the bias is reversed with respect to the previous two simulations: A stimulus is slightly more likely to be seen as a word by the algorithm when it resembles previously presented nonwords more than previously

---

Diependaele & Brysbaert, 2010; Keuleers, Lacey, Rastle, & Brysbaert, under review), we were afraid that even the smallest patterns in the stimuli would be picked up by participants after some time.

presented words. The same is true for nonwords: A stimulus is slightly more likely to be considered as a nonword when it resembles previously presented word rather than previously presented nonwords. The odds are 0.76 to 1, which means that the algorithm is 1.31 times more likely to put a presented stimulus in the other category than the one it belongs to.

Insert Figure 3 here

The likely reason for this change in odds compared to the previous simulations is that, since the nonwords are based on the words in the experiment, LD1NN often finds that a word that served as a template for generating a nonword is among its nearest neighbors, and vice versa. Most nonwords are more similar to one of the words in the experiment than to another nonword. Similarly, most words are more similar to one of the nonwords presented than to another word. To a large extent, Wuggy's default setting of requiring a 2/3 overlap between words and matching nonwords shields against this effect, but is not enough to completely prevent it.

Comparing nonword generation approaches from lexical decision megastudies

In the next section we will look at stimuli from existing experiments and ask to what extent the response probabilities generated by the ED1NN algorithm can predict participant RTs.

We will look at lexical decision data from the English Lexicon project (Balota et al., 2007), the French Lexicon project (Ferrand et al., 2010) and the British Lexicon project (Keuleers et al., under review).

The English Lexicon project

The English Lexicon project's aim was to collect lexical decision data for some 40,000 English words. More than 800 participants each responded to about 3400 stimuli each, yielding about 34 measurements per stimulus. The 40,000 nonwords for the English lexicon project were constructed manually by changing one or two letters in the word stimuli so as to create legal nonwords. Because the nonwords were based directly on words, they were guaranteed to look a lot like words and, therefore, the method, although labor intensive, at first sight seems ideal for the purpose of making a valid lexical decision task. On the other hand, as our simulation with the Wuggy nonwords showed, this way of making nonwords may also create a *reverse* bias, depending on the extent to which the nonwords presented to each participant resemble the words more than the nonwords.

We applied the LD1NN algorithm to the stimuli in the same order as the participants had seen them. Figure 4 shows the result for a randomly chosen participant, which, however, is representative of the pattern observed with all participants. The left panel shows that words are much biased towards nonword decisions, whereas the nonwords are very biased to word decisions. The reason is the same as for the Wuggy algorithm: Because the nonwords were based on the words and were very similar to them, the closest neighbor of a word stimulus mostly was the nonword created from it, and vice versa. The odds for words are 0.34, meaning that word response according to LD1NN is about 3 times less likely when a word is presented than when a nonword is presented.

Needless to say, the bias present in the English Lexicon Project is a rather confusing one, as the information within the stimulus materials goes against the response that

must be made. On the one hand this could slow down participants; on the other hand it could also speed up responses, as even opposite information is information. An algorithm that could make use of this information, for instance, would be based on the rule "provide a different response than the one given to the closest neighbor of this stimulus before".

To find out whether the bias picked up by the LD1NN algorithm also influenced the participants of the ELP, we ran a linear mixed effects model on the correct reaction times for the first 100 participants, using the LD1NN word probability as a predictor and participants as a random effect. Additionally, trial order was included in the analyses to make sure that any effect of LD1NN word probability was not simply an artifact of trial order. For word RTs, the results showed significant effects of trial order (t= 13.73), word probability (t=6.55)[3], and the interaction between both (t=5.39). The estimate for the direction coefficient of word probability was 33.24 (SE=5.07) meaning that, on average, responses slowed down by about 30msec when LD1NN predicted a word probability of 1 compared to when LD1NN predicted a word probability of 0 for a particular stimulus. Thus, participants responded very much like LD1NN would do and took more time to decide that a stimulus was a word when it resembled previously presented words more than when it presented previously presented nonwords. The effect of word probability remained significant (t=3.92) when we added the most important predictors of lexical decision RTs (word frequency and word length) to the model (word frequencies were taken from

[3] P-values are not directly estimable using mixed effects models, but a value of t>2 is often taken as representing a robust significant effect. Estimation with MCMC sampling (Baayen, Davidson & Bates, 2008) showed that all effects reported as significant in this paper, were significant at p<.001.

SUBTLEX-US, Brysbaert & New, 2009). The direction of the coefficient of the effect of word probability stayed in the same order 19.29 (SE = 4.92). For nonwords, the effect of trial order was significant (t = 21.54), as was the effect of word probability (t = 39.67), with its direction coefficient (35.2) indicating slower responses to nonwords with rising word probability. The interaction effect of trial order and word probability could not be included in the analyses for nonwords because of its high correlation (.93) with the main effect of word probability.

The French Lexicon project

In the French Lexicon Project (FLP), Ferrand et al. (2010) collected lexical decision data for 38,840 French words and the same number of nonwords from 975 participants who each responded to 2000 stimuli. The FLP's design was very similar to that of the ELP's, but crucially differed in the way the nonwords were constructed. Monosyllabic nonwords were created by combining onsets and rimes of existing monosyllabic words, and polysyllabic nonwords were created by combining syllables of existing polysyllabic words. In addition, the experimenters made sure that the nonwords matched the words in the experiment on word length, on number of neighbors (defined as the number of words at Levenshtein distance 1), and on the mean, minimum and maximum of the word stimuli's bigram and trigram frequencies.

Since the method for nonword creation in the FLP was so different from that in the ELP, we expected a different picture to emerge when running the LD1NN algorithm on the FLP data. Figure 5 shows the results of that simulation on the stimuli from a random participant in FLP. The biases are far less skewed than in the ELP, and are in

the proper direction. The LD1NN algorithm is more likely to predict word for word stimuli and to predict nonword for nonword stimuli. The logistic regression indicates that LD1NN significantly distinguishes words from nonwords (z=6.2, p < .001), and is 1.6 times more likely to give a word response to a word stimulus than to a nonword stimulus. At the same time, it is important to realize that the data for FLP are based on 2000 trials only, rather than the 3400 trials used in the ELP. As can be seen in the right panel of Figure 5, the average word bias is slightly growing with trial order, so that a like-for-like comparison would be less negative for ELP. Still, based on the analysis with the LD1NN algorithm, the nonword construction method of the FLP seems to be better suited for lexical decision than the one used in the ELP.

We ran linear mixed effects analysis on the RTs of the first 100 participants of the FLP, using the word probability from LD1NN and trial order as fixed effects and participants as random effects. For RTs on words we found an effect of trial order (t=7.73), but failed to observe a significant effect of word probability (t = 0.78) or an interaction between trial order and word probability (t = 1.93). However, when we included word frequency and word length in the analysis, the effect of word probability became significant (t = 2.91), and the direction coefficient (-10.10, SE= 3.47) indicated a speedup of RT with word probability (word frequencies were taken from New, Brysbaert, Veronis, & Pallier, 2007). For nonwords, both the effects of trial order (t = 22.22) and word probability (t=*3.75*) were significant, and the direction coefficient indicated a slowdown of RT with increasing word probability (14.46, *SE*=3.85). So, for FLP the results of the participants are fully in line with the predictions of LD1NN.

Insert Figure 5 here

The British Lexicon project

The British Lexicon project (BLP, Keuleers, Lacey, Rastle, & Brysbaert, under review) presents a radical departure from the earlier methods for collecting lexical decision RTs. In contrast to the ELP and FLP studies, the data in the BLP were collected on a small number of participants who responded to a far larger number of stimuli (28,730 per participant). This method, pioneered by Keuleers, Diependaele, and Brysbaert (2010) increases data collection efficiency and allows for more powerful statistical analyses, but at the same time radically increases the risk that the participants will detect systematic differences between word and nonword stimuli. The authors of the British Lexicon project chose to use the Wuggy algorithm described earlier to generate nonwords, using the following criteria: (1) the nonword matched the syllabic and subsyllabic structure of the target word, (2) it differed from the target word in exactly one subsyllabic segment (onset, nucleus, or coda) for monosyllabic target words and in two subsyllabic segments for disyllabic target words, (3) the transition frequencies of the subsyllabic segments of the target word were matched as closely as possible, and (4) the morphological structure of the word was retained (e.g., if the word was a plural form, we tried to make a matching pseudo-plural).

Figure 6 shows the results of running the LD1NN algorithm on a random participant from the BLP, and shows that there is no detectable bias for nonwords. On average, the nonword stimuli resemble previous word and nonword stimuli to the same

extent. However, the logistic regression showed a significant bias in favor of word responses for word stimuli (z=13, p <.001), with the odds indicating that a word decision was 1.3 more likely when a word was presented than when a nonword was presented. Of all the simulations we performed with LD1NN, this one has the odds ratio closest to 1. It is all the more remarkable that the odds for word decisions remain stable despite the presentation of 28,000 trials.

Insert Figure 6 here

As for the two previous datasets, we performed a linear mixed effects analysis on the BLP RTs, including all 78 participants. For the word RTs, we found a significant effect of trial order (t = 51.80), and LD1NN word probability (t = 12.13), but no significant interaction between these two variables (t= 1.47). Again, the direction coefficient indicated faster RTs to words with higher word probabilities (-13.49, SE= 1.11). Adding frequency and word length to the analysis did not bring any major changes in the magnitude or direction of the effects. The analysis for the nonwords indicated a significant effect of trial order (t = 104.15) and of word probability (t = 16.06), with the direction coefficient (15.87, SE=0.98) indicating slower RTs with higher word probabilities. The interaction effect of trial order and word probability was also significant (t = 7.35).

It is remarkable that despite the high similarity between words and nonwords in the BLP study we still found such statistically strong effects of LD1NN word probability . In this respect, it must be kept in mind that the BLP design enables very powerful analyses (based on a full rectangular matrix of participants x stimuli). Also it must be kept in mind that the participants responded to many more trials than in ELP or FLP.

A final important observation is that, again, the direction coefficients for word probabilities were in the expected direction, with faster RTs for words with higher word probabilities and slower RTs for nonwords with higher word probabilities.

Discussion

In the beginning of this paper, we observed that when the difference between words and nonwords in a lexical decision task can be detected without knowledge of the words in a language, the task in principle does not require lexical access to be solved. Therefore, a bias for word or nonword responses based on systematic differences between the stimuli may be an important factor in an experiment. Indeed, we found that an algorithm without lexical knowledge based on word and nonword similarities (called LD1NN) for various sets of stimuli was able to separate words from nonwords better than by chance.

At this point, it may be remarked that the LD1NN algorithm does require lexical knowledge to provide feedback about the lexical status at the end of each trial (on the basis of which the algorithm learns to discern words from nonwords). This is true and it is in line with the observation that participants usually are aware of the fact that they just made an error. However, this feedback information has more time to come into play than the information on which the decisions are based. In addition, it is imaginable that approaches can be developed that do not require feedback. After all, the algorithm only picks up biases present in the selection of both types of stimuli.

We see the primary application of the LD1NN algorithm as a precautionary measure,

to detect systematic biases in sets of stimuli for new experiments, and to adjust these so that decisions cannot be based on sublexical information. As the interest in lexical decision experiments for the establishment of re-usable databases grows, and as experiments get longer, such precautions become more and more important. Evidently, LD1NN can also be used to analyze stimuli from existing experiments to highlight potential difficulties and to guide re-analysis.

We showed that different ways of making nonwords yield different response biases. As probably every seasoned experimenter knows, using random letter strings as nonwords is not advisable, as the words are very easily detectable even without much knowledge of the words. More surprisingly, we showed that very much the same biases were present for random samples of nonwords from an existing database of legal nonwords. As a matter of fact, these nonwords led to even greater biases, because there was not only information present in the word stimuli but in the nonword stimuli as well (given that the nonwords tended to resemble the other nonwords more than the words). Additionally, we showed that creating nonwords by changing one or two letters in the word stimuli of the experiment leads to another difficulty, because the similarity across response categories may become larger than the similarity within categories. Nonword generating approaches such as Wuggy (Keuleers et al., 2010) or the statistical approach used for FLP lead to less biased stimuli. Of these approaches, Wuggy may be the more convenient. It is currently available for eight languages (Dutch, French, English, Basque, Spanish, Serbian, German, and Vietnamese). In combination with the LD1NN algorithm to test the final composition of a set of stimuli, we expect that it can be used to design experiments

with almost no inherent bias left.

We do not claim that the LD1NN algorithm is a cognitive model of the way in which participants detect systematic differences between words and nonwords. We simply argue that in many lexical decision experiments there is much more sublexical information participants may exploit than acknowledged by the experimenters. In addition, the algorithm may not be so far-fetched given that it bears close resemblances to exemplar based models in psychology and to memory-based language processing methods (Daelemans & van den Bosch, 2005), which have been successfully applied in the domain of computational linguistics. The fact that, in all the experiments we tested, a simple algorithm produced probabilities that were predictive of lexical decision performance suggests that inherent bias may account for more variation in response measures than we are currently aware of. In this respect, it is important to notice that the relations between word probabilities and RTs in existing megastudies were consistent with the LD1NN predictions, with biases against word decisions leading to slower RTs on word trials and to faster decisions on nonword trials, except in ELP, where the bias was opposite to the response to be given.

Conclusion

We presented a novel method to detect inherent bias in lexical decision experiments. The LD1NN algorithm can be used to detect bias, and to guide stimulus construction. Analysis of mock datasets and existing lexical decision experiments indicated that (1) inherent bias in a lexical decision experiment is very hard to avoid, (2) traces of bias are reliably detected when behavioral data are analyzed, and that

(3) some nonword generation approaches lead to less biased stimulus sets. In particular, the nonword generation algorithm used by the Wuggy program (Keuleers & Brysbaert, 2010) seems to be the best option for constructing lexical decision experiments with little inherent bias. Wuggy is available at http://crr.ugent.be/Wuggy. The LD1NN algorithm is available from http://crr.ugent.be/LD1NN and includes R source code to run simulations and to plot figures such as the ones presented in this paper.

References

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, *59*(4), 390–412.

Balota, D. A., Yap, M. J., Cortese, M. J., Hutchison, K. A., Kessler, B., Loftis, B., Neely, J. H., et al. (2007). The English lexicon project. *Behavior Research Methods*, *39*(3), 445-459.

Daelemans, W., & van den Bosch, A. (2005). *Memory-based language processing*. Cambridge University Press.

Ferrand, L., New, B., Brysbaert, M., Keuleers, E., Bonin, P., Méot, A., Augustinova, M., et al. (2010). The French Lexicon Project: Lexical decision data for 38,840 French words and 38,840 pseudowords. *Behavior Research Methods*, *42*(2), 488.

Fix, E., & Hodges, J. L. (1951). *Discriminatory analysis, nonparametric discrimination. USAF School of Aviation Medicine, Randolph Field, Tex*. Project 21-49-004, Rept. 4, Contract AF41 (128)-31.

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 Psychologia, 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 and performance, 24*, 1162-87.

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

Keuleers, E., Diependaele, K., & Brysbaert, M. (2010). Practice effects in large-scale visual word recognition studies: A lexical decision study on 14,000 Dutch mono-and disyllabic words and nonwords. *Frontiers in Psychology*, *1*, 1.

Keuleers, E., Lacey, P., Rastle, K., & Brysbaert, M. (under review). The British Lexicon Project: Lexical decision data for 28,730 monosyllabic and disyllabic English words.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady* (Vol. 10, pp. 707–710).

New, B., Brysbaert, M., Veronis, J., & Pallier, C. (2007). The use of film subtitles to estimate word frequencies. *Applied Psycholinguistics, 28*(4), 661–677.

Rastle, K., Harrington, J., & Coltheart, M. (2002). 358,534 nonwords: The ARC nonword database. *The Quarterly Journal of Experimental Psychology Section A*, *55*(4), 1339–1362.

Appendix: R code for the LD1NN algorithm

```
LD1NN <-function(stimulus, type, reference.level){
    n<-length(stimulus)
    bias<-rep(0,n)
    for(i in 2:n){
        distances<-levenshteinDist(stimulus[i],stimulus[1:i-1])
        unique.distances<-sort(unique(distances))
        minimum.distance<-unique.distances[1]
        indexes<-which(distances<=minimum.distance)
        distribution<-type[1:i-1][indexes]
        probability<-sum(distribution==reference.level)/length(distribution)
       bias[i]<-probability-(1-probability)
    }
    return(bias)
}
```

Table 1. LD1NN, applied to a set of 10 randomly chosen stimuli. The first column shows the current stimulus in the experiment, along with its lexicality (Word=W, Nonword=N). The second column shows the previous stimuli in the experiment, ordered by their Levenshtein distances from the current stimulus. The third column shows the Word probability (p) for the current stimulus, or the ratio of number of words at the nearest distance to the total number of stimuli at the nearest distance. The final column shows the Word bias (2p-1) for the current stimulus, with 1 and -1 indicating complete bias for and against a Word response.

Current stimulus	Levenshtein distances for previous stimuli	Word probability	Word bias
alcirans (N)	-	0/0	0
walfine (N)	[5] alcirans (N)	0/1	-1
doller (N)	[6] walfine (N); [7] alcirans (N)	0/1	-1
cowshed (W)	[5] doller (N); [7] walfine (N); [8] alcirans (N)	0/1	-1
dredge (W)	[5] doller (N); [6] walfine (N), cowshed (W); [8] alcirans (N)	0/1	-1
tidid (N)	[5] dredge (W); [6] walfine (N), doller (N), cowshed (W); [7] alcirans (N)	1/1	1
peahens (W)	[6] alcirans (N), walfine (N), doller (N), cowshed (W), dredge (W); [7] tidid (N)	2/5	-0.2
unnoon (N)	[6] walfine (N), doller (N), dredge (W), tidid (N), peahens (W); [7] alcirans (N), cowshed (W)	2/5	-0.2
prea (N)	[4] dredge (W); [5] doller (N), tidid (N), peahens (W); [6] cowshed (W), unnoon (N); [7] alcirans (N), walfine (N)	1/1	1
sown (W)	[4] unnoon (N), prea (N); [5] doller (N), cowshed (W), tidid (N); [6] walfine (N), dredge (W), peahens (W); [7] alcirans (N)	0/2	-1

Figure 1. Results of LD1NN algorithm presented with a group of 2500 mono- and disyllabic English words and 2500 random letter strings, matched for length. 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.

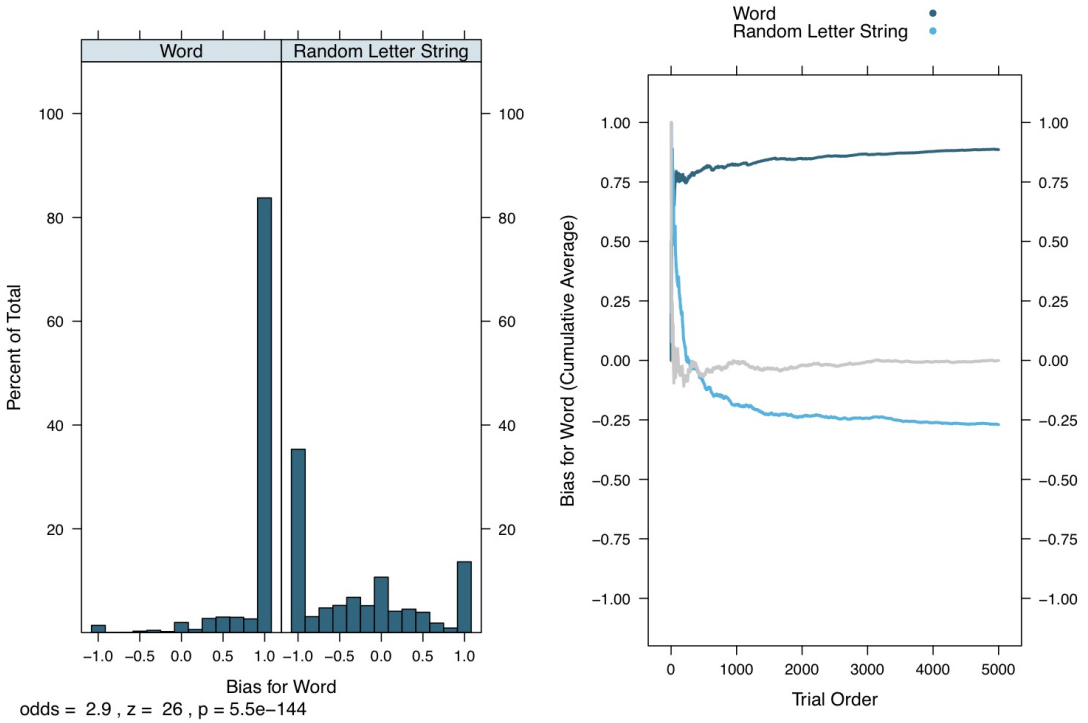

Figure 2. Results of LD1NN algorithm presented with a group of 2500 mono- and disyllabic English words and 2500 nonwords from the ARC nonword database, matched for length. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

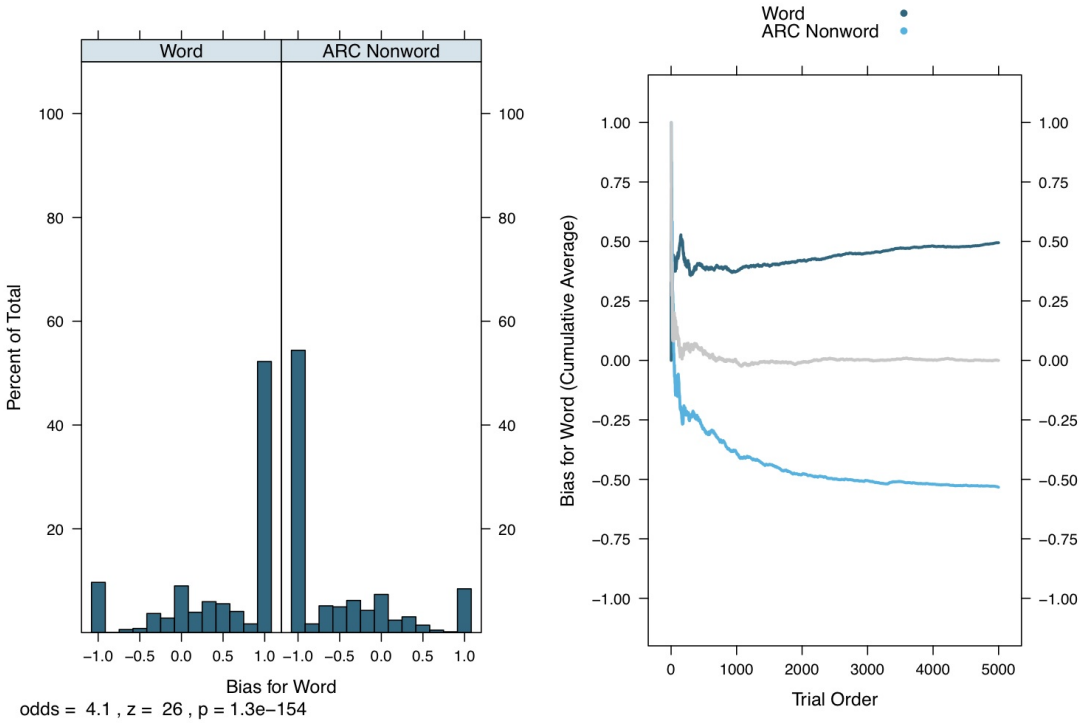

Figure 3. Results of LD1NN algorithm presented with a group of 2500 mono- and disyllabic English words and 2500 words generated by Wuggy, matched for length. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

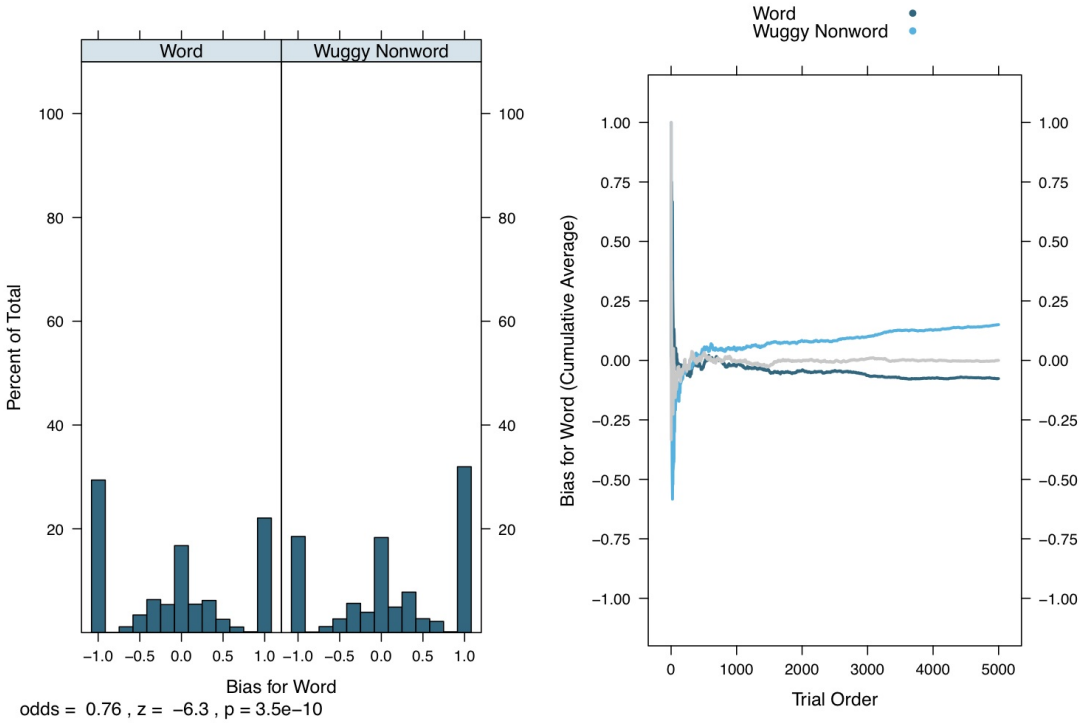

Figure 4. Results of LD1NN algorithm presented with the stimuli presented to a random participant in the English Lexicon project. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

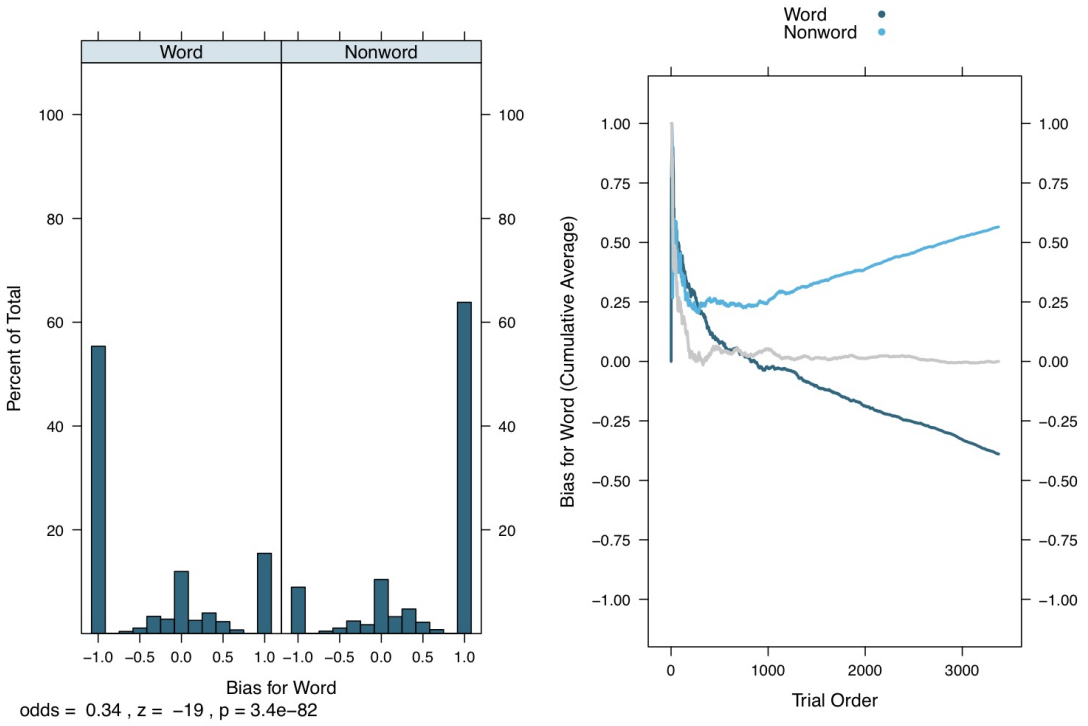

Figure 5. Results of LD1NN algorithm presented with the stimuli presented to a random participant in the French Lexicon project. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.

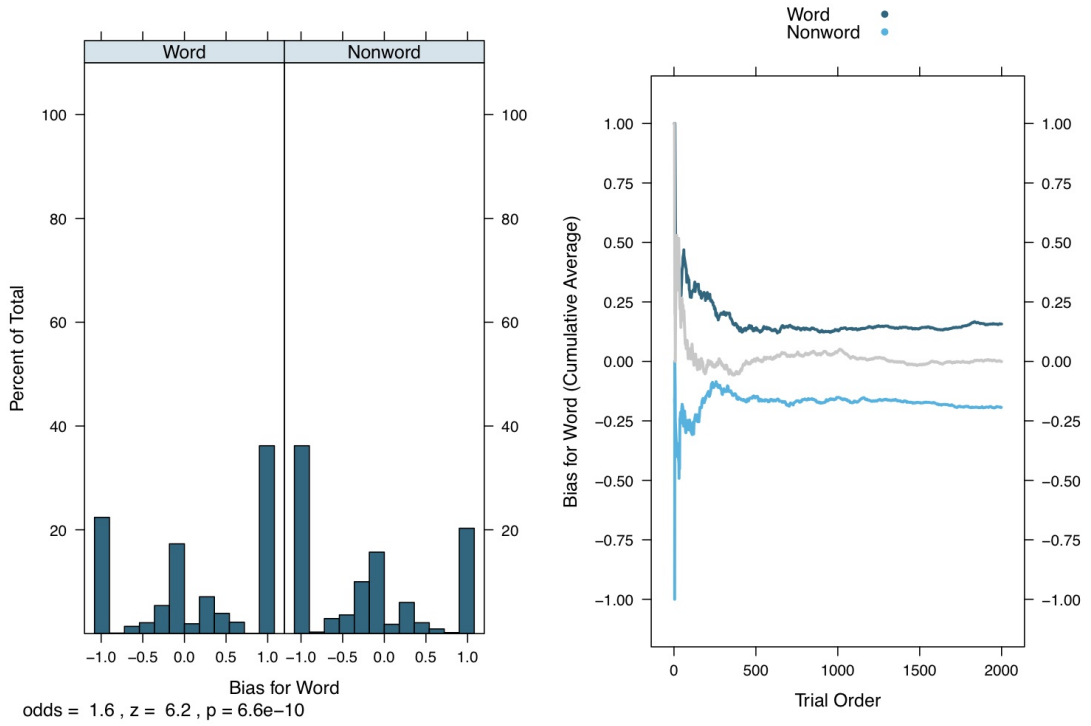

Figure 6. Results of LD1NN algorithm presented with the stimuli presented to a random participant in the British Lexicon project. Left panel: distribution of word bias for both words and nonwords. Right panel: cumulative average of word bias for words and nonwords; the grey line indicates the numerical bias for words based on the percentage of words vs. nonwords processed up to that point.