Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 172–181
Marseille, 11–16 May 2020

Orthographic Codes and the Neighborhood Effect: Lessons from Information Theory

Stéphan Tulkens, Dominiek Sandra, Walter Daelemans
CLiPS - Computational Linguistics Group
Department of Linguistics
University of Antwerp
{firstname.lastname}@uantwerpen.be

Abstract

We consider the orthographic neighborhood effect: the effect that words with more orthographic similarity to other words are read faster. The neighborhood effect serves as an important control variable in psycholinguistic studies of word reading, and explains variance in addition to word length and word frequency. Following previous work, we model the neighborhood effect as the average distance to neighbors in feature space for three feature sets: slots, character ngrams and skipgrams. We optimize each of these feature sets and find evidence for language-independent optima, across five megastudy corpora from five alphabetic languages. Additionally, we show that weighting features using the inverse of mutual information (MI) improves the neighborhood effect significantly for all languages. We analyze the inverse feature weighting, and show that, across languages, grammatical morphemes get the lowest weights. Finally, we perform the same experiments on Korean Hangul, a non-alphabetic writing system, where we find the opposite results: slower responses as a function of denser neighborhoods, and a negative effect of inverse feature weighting. This raises the question of whether this is a cognitive effect, or an effect of the way we represent Hangul orthography, and indicates more research is needed.

Keywords: Cognitive Methods, Lexical Database

1. Introduction

One of the core issues in contemporary models of word reading is the representation of orthography. Orthography, in this case, can be understood as the visual information of a word as it is represented by the brain. It therefore does not generally refer to the actual visual presentation of a word, such as the font and case, but to a more abstract visual representation (Dehaene et al., 2005; Dehaene, 2009). One of the reasons orthography is such a core issue is that older models, such as the Interactive Activation (McClelland and Rumelhart, 1981) and DISLEX (Miikkulainen, 1993; Miikkulainen, 1997) models, assumed that words were represented as an array of slots, an assumption that has recently been shown to be false. As readers are remarkably flexible at decoding orthographic information from jumbled strings (Schoonbaert and Grainger, 2004; Perea et al., 2008), the positional information in orthographic representations has to be flexible to some degree; if this were not the case, readers would not be able to decode transposition neighbors, e.g., JUGDE - JUDGE, efficiently. These, and other, phenomena have motivated the search for various models or featurizations of orthography that can accurately capture empirical data, which has also been dubbed the search for an orthographic code (Grainger, 2008; Grainger, 2018). Examples of such feature sets, or orthographic codes, are character ngrams, or wickelgraphs (Wickelgren, 1969), fourteen segment coding (Rumelhart and Siple, 1974), slot-based coding (McClelland and Rumelhart, 1981), and various open bigram schemes (Whitney, 2001; Schoonbaert and Grainger, 2004; Whitney and Cornelissen, 2008).

Another issue related to the representation of orthography is the discovery that orthographic similarity plays an important role in how quickly words are identified; words that look more like other words are, generally, identified more quickly (Coltheart, 1977; Andrews, 1989; Grainger, 1990; Yarkoni et al., 2008; Perea, 2015). Orthographic codes, or feature sets, thus have an important role; they constrain the inferences models can make about the similarity of words, while also defining the orthographic similarity between words. While many feature schemes have been contrasted (Davis and Bowers, 2006; Grainger, 2008; Davis, 2010; Kinoshita and Norris, 2013), there is little consensus about how words are represented, or which orthographic code is to be preferred over the other. In this paper, we contrast various features on their ability to explain variance in reaction times on lexical decision tasks, and show that optimizing these feature sets leads to gains in explained variance.

1.1. The neighborhood effect

We use the neighborhood effect as a task on which to test the fit of orthographic codes. As described above, the neighborhood effect is the effect that words which have more orthographic similarity to other words are read faster (Andrews, 1989; Perea, 2015). It is often thought to involve effects related to co-activation between word representations, a theoretical position mainly inspired by the Interactive Activation model (McClelland and Rumelhart, 1981). That is, the neighborhood effect elicited by a word such as PLEAD is a function of the orthographic similarity of PLEAD to other words in the lexicon. Because the similarity of a space is dependent on the features used, changing feature sets or feature weighting will then also impact the neighborhood effect. For example, if a feature set explicitly models letter order, PLEAD and LEAD have zero similarity, because they do not share any letters in any position. Similarly, if a feature set allows for transposition, words such as COLD and CLOD will have higher similarity than in feature sets that do not model transposition. Hence, the assumption that the neighborhood effect is highly dependent on orthographic similarity gives us a way of objectively evaluating different feature sets. Feature sets whose neighborhoods explain more variance are, in our reasoning, more plausible.

1.2. Main Contributions

In this paper, we analyze three different feature sets and their various parameterizations, and show that the way these feature sets are currently used is suboptimal, for two reasons: first, the parameterizations of the feature sets used in other papers are not ideal. Second, the number of nearest neighbors considered in calculating the neighborhood effect is not ideal. Furthermore, we show that the optimal parameters for these metrics differ from the parameters used in psycholinguistic research on word reading. We directly compare our results to previous work, i.e., Tulkens et al. (2018a), and show that the conclusions drawn from this work are incomplete because the research was carried out with a suboptimal number of nearest neighbors and parameters. We thus conclude that many different orthographic codes are equally feasible as far as the neighborhood effect is concerned, but only when properly optimized, which was not the case before. In a second experiment, we show that the inverse of mutual information weighting improves the explained variance of the neighborhood effect for all feature sets, and that weighing with regular mutual information almost removes the neighborhood effect. This implies that features that frequently occur with a smaller set of words are less important for calculating the neighborhood effect. We show that these features generally correspond to bound morphemes, such as plural suffixes. Additionally, we present the first results on modeling the orthographic code of Korean, a non-alphabetic language.

2. Materials and Methods

In this section, we introduce the metric we use to measure the similarity of words in feature space, the features, and the corpora used in this study.

2.1. Metrics for Measuring Neighborhoods

The standard metric for measuring neighborhoods is OLD20 (Yarkoni et al., 2008), which is defined as the mean Levenshtein distance (Levenshtein, 1966) to the 20 closest neighbors. As OLD20 operates on string representations, alternative feature representations can not be easily adapted to, or incorporated in, OLD20. Features sets, on the other hand, are more flexible, and can represent, for example, discontinuous regularities in words, transpositions, alignment, or lack thereof (Whitney, 2001). To bridge the neighborhood effect and string metrics, Tulkens et al. (2018a) introduced rd, a metric that calculates the neighborhood density for arbitrary feature spaces. Mathematically, rd is similar to OLD20 (Yarkoni et al., 2008), as it is the sum of the cosine distances to the k closest featurized neighbors.[1]
Following Tulkens et al. (2018a), rd is described as follows:

$$\mathrm{rd}(x, X, k) = \sum_{i=1}^{k} \cos(x, X)^i \qquad (1)$$

Where $\cos$ is the cosine distance, x is featurized item, and X is the set of all featurized items, which may or may not include x. We assume that the output of $\cos$ is sorted, so that the function returns the sum over the k closest items. In their original experiments, Tulkens et al. (2018a) set k to 20, to explicitly compare to OLD20. In this paper, we relax this requirement, and investigate whether different values of k work better for different neighborhoods. Because Tulkens et al. (2018a) showed that rd outperformed OLD20, and because OLD20 can not be feature weighted in Experiment 2, we leave it out of the current discussion.[2]

2.2. Featurizations and Their Parameterizations

We use several existing feature sets, all of which have been implemented in Wordkit (Tulkens et al., 2018b). An overlooked aspect of these feature sets in psycholinguistic research is that almost all of them are amenable to parameterization. As an example, consider the well-known open bigram encoding (Schoonbaert and Grainger, 2004; Grainger and Van Heuven, 2004; Whitney, 2001). There is no reason, be it computational, cognitive, empirical, or otherwise, to restrict ourselves to using bigrams instead of, say, trigrams. We show that, for most implemented feature sets, there exist multiple parameter settings which can be explored.

2.2.1. Slot-based Encoding

A slot-based encoding consist of orthogonal vectors, aligned in slots. Such an encoding is identical to the letter layer used in the original Interactive Activation model (McClelland and Rumelhart, 1981). This encoding assumes that all characters are completely orthogonal, in the sense that any character is equally different to any other character, regardless of its visual characteristics. Research has shown that transposition (Perea and Lupker, 2004; Perea et al., 2008) and deletion (Van Assche and Grainger, 2006) neighbors are more easily decoded than substitution neighbors. As slot-based codes assigns equal similarity to transposition and substitution neighbors, e.g., $\mathrm{sim}(\text{PAWN}, \text{PWAN}) = \mathrm{sim}(\text{PAWN}, \text{PXYN})$, slot-based codes are thought to be insufficient. Nevertheless, Tulkens et al. (2018a) showed that this encoding can still account for the most explained variance when its neighborhood is entered into a linear regression.

2.2.2. ngrams

We also consider character ngrams, also known as wickelgraphs (Wickelgren, 1969) as a feature set. The wickelgraph encoding decomposes a word into a set of character ngrams, where n is a free parameter. ngrams do not encode order, which causes words that are embedded in other words, e.g. PAN and SPAN, to still share some similarity. All words are padded before calculating the ngrams, because otherwise words shorter than n can not be accurately featurized. In all our experiments, we use $n \in \{2, 3, 4\}$.

[1] Although OLD20 uses the mean distance, instead of the sum, this does not have any bearing on the fit to the data, as the denominator in the mean is k for all items. The sum has the further advantage of not arbitrarily leaving out the item itself; which is required in the definition of OLD20.

[2] We did carry out experiments, and separately optimized k, using OLD, but this did not show anything interesting; rd using one hot encoded strings still outperforms OLD

	FR	SP	NL	UK	US
# words	38,335	44,853	24,530	28,480	30,639

Table 1: The number of words left in each of the corpora after preprocessing.

		FR	SP	NL	UK	US
ngram	n	4	4	4	4	4
Open ngram	n	3	3	3	3	3
	w	3	3	3	4	3

Table 2: The optimal parameters for the two feature sets that had parameters to optimize. Note the lack of variation across parameterizations.

	FR	SP	NL	UK	US
Slots	6	8	4	2	6
ngram	4	6	5	5	7
Open ngram	3	6	5	2	6

Table 3: The optimal values of k for the optimized models.

Note that we explicitly model ngrams as multisets, and the generated vectors thus contain the count of each feature. This is necessary, because otherwise ngrams would not be able to represent the difference between, for example, BANANA and BANANANANA, as these contain the same ngram types, but in different frequencies.

2.2.3. Constrained Open ngrams

The constrained open ngram encoding (Schoonbaert and Grainger, 2004) is a refinement of the open ngram encoding (Whitney, 2001; Grainger and Van Heuven, 2004). An open ngram encoding is defined as the set of n-combinations of letters a word, where the letters within a combination are ordered by their occurrence in the word. That is, the word SWAN generates the following open 2-gram (or bigram) features: {SW, SA, SN, WA, WN, AN}. One issue with the original open ngram encoding is that it does not put a constraint on generated combinations, which is an issue, both theoretical and practically. As an example, the word QUARANTINE, in the unconstrained setting, generates 45 bigrams, many of which are separated by more than 4 intervening letters. Because of these bigrams with wide gaps, performance with unconstrained open ngrams tends to suffer.[3] To remedy this, constrained open ngrams were introduced; these only allow for the construction of an ngram with some pre-specified window, which we call w.

We use $n \in \{2, 3\}$ and $w \in \{2, 3, 4, 5\}$, with the added constraint that w must be smaller than n. When $w = (n-1)$, constrained open ngrams reduce to regular ngrams. Constrained open ngram encoding leads to more parsimonious representations, especially for larger values of n, as the number of possible ngrams decreases because of the window constraints. Like the ngrams above, we also represent these as multisets, and add padding.

[3]This is also what we observed in our experiments, where regular open ngrams performed far worse than their constrained alternatives.

2.3. Corpora

Following previous research into the neighborhood effect, e.g., Yarkoni et al. (2008), we measure the neighborhood effect using Reaction Times (RT) on a lexical decision task, while controlling for log frequency and length, two other control variables. We use various corpora from five different languages: British English, US English, Dutch, French, and Spanish. In all cases, we start from a megastudy of lexical decision RT measurements, to which we then add frequency measurements from subtitle corpora, if necessary. For all corpora, we apply the following preprocessing steps: first, we remove all words which are not lower-cased, and words without RT or frequency measurements. We remove the non-lower-cased words because not all databases have mixed-case words, and because we have no easy way of determining what to do uppercase letters with regards to the neighborhood effect. Second, we remove all words which contain non-alphabetic characters, such as the genitive marker, space, or dash.

The American English corpus was constructed using the English Lexicon Project (ELP) (Balota et al., 2007) and the the SUBTLEX-US database (Brysbaert and New, 2009). The British English corpus was constructed using the British Lexicon Project (BLP) (Keuleers et al., 2012) and the SUBTLEX-UK database (Van Heuven et al., 2014). The French corpus was constructed from the Lexique database (New et al., 2001; New et al., 2007), and the French Lexicon Project (Ferrand et al., 2010). The Dutch corpus was constructed using the Dutch Lexicon Project (DLP) (Keuleers et al., 2010b) and SUBTLEX-NL (Keuleers et al., 2010a). Finally, the Spanish corpus was constructed from the SPALex database (Aguasvivas et al., 2018). Because SPALex already includes frequency counts, we did not need an auxiliary corpus for frequency counts. Note that, while SPALex involves lexical decision, the task used in the construction of that corpus did not explicitly ask participants to respond as quickly as possible. Nevertheless, the results can be reinterpreted as being largely equivalent to those of lexical decision (Aguasvivas et al., 2018).

For the French and Spanish corpora, we had to decide whether to keep or remove words that contained letters with diacritic markers, such as TRÉS, as the status of these markers with regard to their orthographic decomposition is unclear. As the removal of diacritic markers led to the loss of a lot of words, and created the conundrum of us having to decide what to do with duplicate forms, we chose to keep them.

3. Experiment 1: Optimization

In the first experiment, we consider all possible featurizations and their parameterizations. We then use cross-validation to estimate the optimal parameters for the various feature sets we have. Similarly, we also use the same search procedure to estimate the optimal number of nearest neighbors for each feature set. The search procedure is performed on a training set, which consists of 90% of our data. The other 10% is held out as a test set, and is only used to test our final models, and not in any of the optimization procedures. Both the cross-validation and initial train-test split were stratified by length and binned log frequency.

	Slots		**ngrams**		**Open ngrams**	
	Reg	**Opt**	**Reg**	**Opt**	**Reg**	**Opt**
FR	*.391*	**.402**	.371	.390	.355	.386
SP	*.600*	**.603**	.590	.595	.572	.588
NL	*.321*	**.324**	.307	.322	.283	.310
UK	.356	**.367**	*.357*	.360	.350	.366
US	*.506*	.506	.501	**.509**	.481	.501

Table 4: The test R^2 on the base and optimized models. The R^2 scores in *italics* denote the best unoptimized (regular) models, while the **bold-faced** scores denote the best models for each language.

3.1. Model Selection

As noted above, RD has a parameter, k, that determines the number of nearest neighbors to take into consideration. In previous work, e.g., Yarkoni et al. (2008) and Tulkens et al. (2018a) and the papers that deploy these metrics as control variables, k has been set to 20. As we do not know what the effect of k is for our different feature sets, we use cross-validation and held-out test-data to search for the optimal value of k for each feature set on each dataset. For all ngram feature sets, we also use the same cross-validation loop to find the best feature set parameters, e.g., the optimal value of $window$ and n. We use a 90% - 10% train-test split, stratified by log frequency and length. Then, for each model, we perform 10-fold cross-validation, again stratified by log frequency and length, on the training set to estimate the best k and other parameters. Within each fold, we fit the following regression model:

$$\text{rt} \approx \log_{10}(\text{frequency}) + \text{length} + \text{rd} + \epsilon$$

Where rd is the representation distance from each word to the k closest neighbors, as detailed above. Using the fit model, we calculate the R^2 on the held-out data of that fold. We compare all our models to a baseline model:

$$\text{rt} \approx \log_{10}(\text{frequency}) + \text{length} + \epsilon$$

That is, the model above but without the neighborhood effect added in.

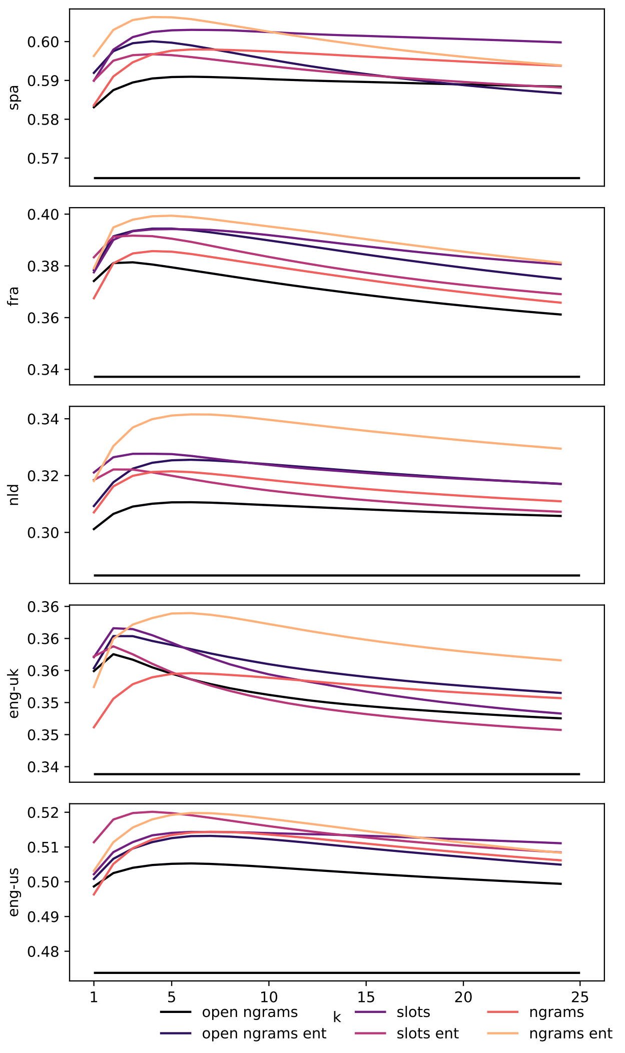

Figure 1: The R^2_{adj} scores of the optimized models on the full dataset over different values of k. The featurizations ending with ent are the feature weighted counterparts of the regular feature sets. The black line indicates a simple regression model with only length and log frequency as predictors.

3.2. Results

For each feature set, we selected the version with the lowest mean test R^2 over all folds, and ran this model on the test data. As noted above, we jointly select k and the optimal parameters for both feature sets. The selected parameters are listed in Table 2, while the optimal values of k are listed in Table 3. The distribution over R^2_{adj} scores for all featurizers, and all values of k over all languages, is shown in Figure 1. Both outcomes provide a stark contrast with the way neighborhood metrics have been deployed so far. First, as mentioned above, all metrics so far tended to use a k of 20; as the values in the Table indicate, this is far from optimal, as all the values are lower than 10, and sometimes as low as 2, which indicates that 20, as an arbitrary number, was far too high. Second, the optimal parameters for the ngrams and open ngrams are also different from those typically deployed in the literature, and are consistent across languages. That these parameters converge across languages provides strong evidence in favor of the proposed optimizations: feature sets and models should be optimized and cross-validated on a diverse set of data to obtain support for specific parameters. The R^2 scores of the regular and optimized models are shown in Table 4, which shows that the predictive power of all models increases with optimization, although much more so for the ngrams and open ngrams than the slot-based features. To confirm this pattern, we bootstrapped the differences between R^2 scores over 10,000 samples (Efron and Tibshirani, 1994), which allows us to compare distributions over the differences using parametric statistics, such as a T-test. Paired T-tests revealed that differences were significant at $p < .0001$, indicating that all optimized models significantly outperformed the baseline models, even when correcting using Bonferroni correction (Bonferroni, 1936). This shows that the feature sets that have been in use so far were probably suboptimal.

4. Experiment 2: Entropy

In a second experiment, we investigate the impact of weighing individual orthographic features. While weighted open bigrams (Whitney and Cornelissen, 2008) introduce a weighting scheme based on the number of intervening letters, there has been very little work on applying feature weighting to the neighborhood effect. Feature weighting has a long history, and has been shown to be effective when employed with kNN models in a Memory-Based Language Processing framework (Daelemans and Van den Bosch, 2005). In the TIMBL toolkit (Daelemans et al., 1998), for example, the IGtree algorithm (Daelemans et al., 1997) uses the information gain (IG) feature weighting metric to construct a tree, which can then be used to more efficiently select the neighbors relevant for classification. Other examples of such feature weighting techniques are MVDM (Stanfill and Waltz, 1986), which weights distances based on the similarity between two features. As the neighborhood models we use are *de facto* kNN models, we investigate whether feature weighting is an effective way of increasing performance of our models.

4.1. Entropy weighting

We use entropy (H) as a feature weighting scheme. For a discrete probability distribution, entropy (H) is given by the following equation:

$$\mathrm{H}(X) = \sum_{i=1}^{|X|} -P(X_i)\log_2(P(X_i)) \qquad (2)$$

Calculating the entropy separately for each column in a D-dimensional feature matrix leads to a vector of weights, $W^{\mathbb{R}^D}$, which we use to weigh the features. As we only use binary feature indicators, each feature can either occur or not occur, low entropy values are thus associated with a feature occurring with no words, or with all words. The highest entropy value occurs only when a feature occurs in exactly half of all types in our corpus, i.e., when the distribution is uniform. We use entropy over the more common Mutual Information (MI), which also takes class distributions into account, because in our case the class distribution is uniform.

During experimentation, we found that using entropy as a feature weighting measure completely removed any neighborhood effect, and reduced any linear regression models to baseline level. Hence, we hypothesized that one of the drivers of the neighborhood effect is the complement of entropy.

Two candidates for such a complement are extropy (Lad et al., 2015), and negentropy (J) (Schrödinger, 1944; Brillouin, 1953). As extropy and entropy are identical for binary values (Lad et al., 2015), we experiment with negentropy. In the continuous case, negentropy is defined as the difference in entropy between the distribution and the entropy of a normal distribution with the same mean and variance (Brillouin, 1953). Note that, for continuous distributions, the normal distribution leads to a maximized entropy. As such, negentropy is always non-negative, and 0 if and only if the distribution is normal. Hence, in accordance with the definition for continuous distributions, for a D-dimensional discrete distribution, we define negentropy as the difference between the entropy of the D-dimensional uniform distribution, which is the situation in which entropy is maximized, and the entropy of the distribution. As such, negentropy over a discrete distribution always non-negative, and 0 if and only if the distribution is uniform, leading to the same constraints as continuous negentropy. If $\mathbb{U}$ is the uniform distribution for a given dimensionality, then negentropy is defined as follows:

$$\mathrm{J}(X) = \mathrm{H}(\mathbb{U}) - \mathrm{H}(X) \qquad (3)$$

We use the same experimental setting as before. Instead of separately optimizing the entropy-weighted feature sets, we simply apply the entropy functions to the best-performing models in cross-validation, and contrast these to the models on the test set. We also apply regular entropy, and investigate whether the scores of neighborhoods are equal to the baseline.

Eng-US		**Fra**	
trigram	J	**trigram**	J
s##	0.260	s##	0.148
e##	0.411	e##	0.220
d##	0.440	t##	0.288
##s	0.463	es#	0.457
ed#	0.507	nt#	0.467
##c	0.530	ent	0.470
ing	0.545	##c	0.504
y##	0.564	##r	0.547
g##	0.570	##p	0.549
ng#	0.581	##a	0.577

Table 5: The top 10 trigrams with the lowest negentropy for US English and French, respectively. Notice how most of these consist of padding ngrams, corresponding to common bound morphemes and word endings, such as ‘ing’ in English and ‘en’ in French.

4.2. Results

The results are listed in Figure 2, which shows the distribution of R^2_{adj} bootstrapped over 10,000 samples. As the Figure shows, the models weighted with H are near baseline performance, while the models weighted with J almost all outperform their non-weighted counterparts. We confirmed this by calculating the 95% CI of differences between the weighted and non-weighted variants of the feature sets. This again led to significant differences between all optimized feature sets and their feature weighted counterparts; for all feature sets except the slot-based feature set, negentropy weighting outperformed entropy weighting. For slot-based feature sets, only the US English slot-based feature set outperformed the optimized counterpart. In terms of absolute performance, the negentropy weighted feature set showed the highest performance in all cases, thus questioning the earlier conclusion of Tulkens et al. (2018a) that one hot encoded representations are the best performers. As Table 5 shows, negentropy, when applied to charac-

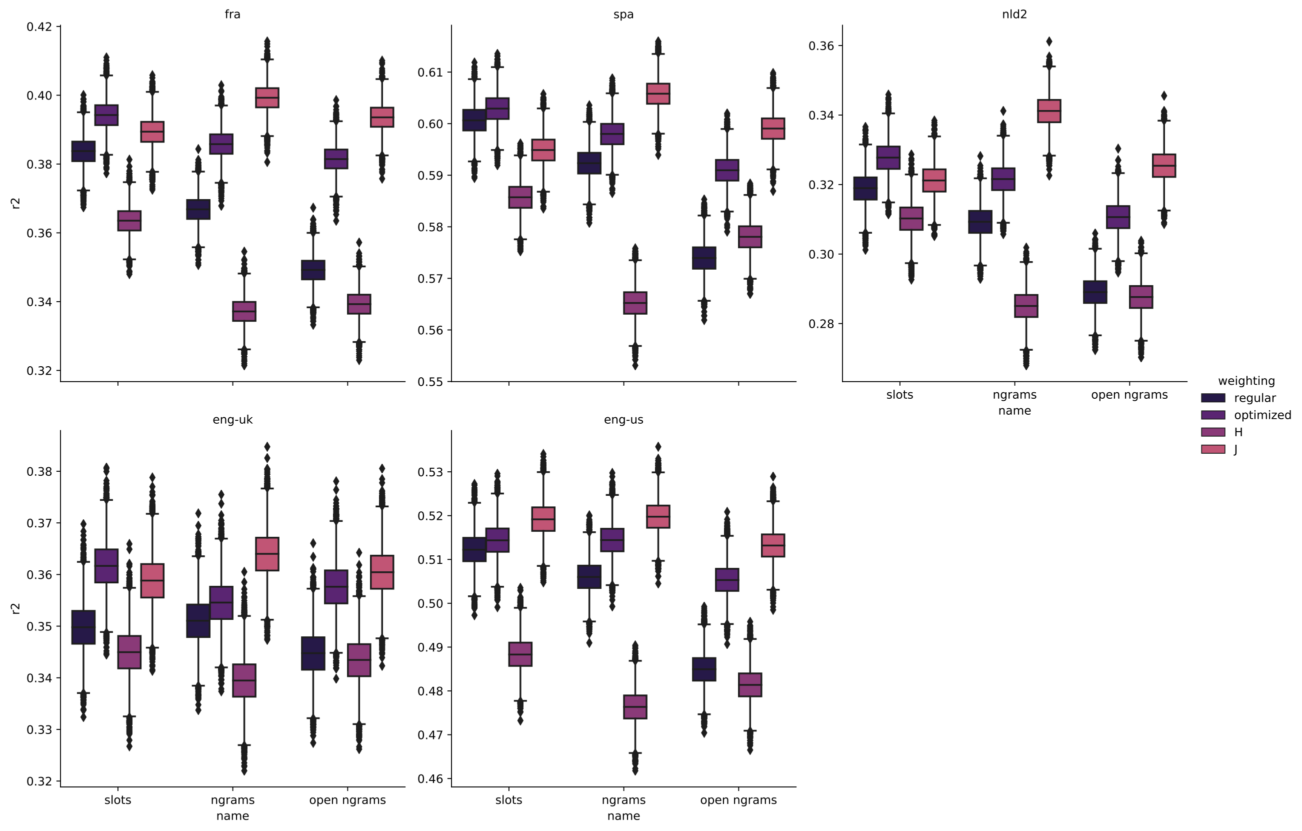

Figure 2: The distribution over R^2_{adj} on all data for all feature sets in their regular, optimized, weighted, and inversely weighted variants. The plots shows an effect of weighting, especially for the ngrams models. For all feature sets in all languages, the weighted variant is low, and near baseline level.

ter trigrams, assigns lower weights to common word endings and bound morphemes, i.e., the '-s' and '-ed' suffixes marking the simple and perfect past, and '-ing' marking the present continuous for English, and the '-nt' and 'ent' suffixes, marking the third person plural in French. We observe similar results for the other languages, where, for example, the morphemes '-en' and '-ado' are weighted down heavily for Dutch and Spanish, respectively. The only models for which the weighting consistently does not improve performance are the slot-based models. An explanation for this lies in independence of the one hot encoded representations; using ngram features, it is very clear whether a given feature is a suffix or not, because it occurs at the end of a word, no matter the length of that word. Slot-based features on the other hand, do not have information about the length of a word, and assigns the same representation to the letter S, e.g., in WALKS and the letter S in EURASIA. Note that the lower weights for bound morphemes can only be related to their relative frequency of occurrence, not their status as morphemes, as feature weighting has no information about morphological segmentation. The frequency of occurrence of these morphemes thus has to be the driver of the weighting. What is peculiar, however, is that the down-weighting of morphemes leads to better performance across all languages. Whether this is an effect of letter entropy or frequency, or actually related to morphological processing is an open question.

5. Experiment 3: Korean

We now apply the strategy and methodology of the previous two experiments to Korean. As argued by Frost (2012), the results of experiments such as the one we performed above carry with them a bias towards alphabetic writing systems, in which it is easy to confuse and transpose letters. In Arabic and Hebrew, for example, which have more rigid position coding, transposition effects and substitution effects are processed differently, and do not lead to the strong priming effects seen in alphabetic languages (Velan and Frost, 2007; Velan and Frost, 2009; Perea et al., 2010; Velan et al., 2013). The Korean alphabet, also called Hangul, is interesting in this regard, as it has alternatively been characterized as featural (Sampson, 1985), syllabic, and alphabetic (Pae, 2011). Like an alphabetic writing system, words are made up out of letters, and are separated by spaces. The letters, however, correspond to syllables instead of phonemes, as in a syllabic writing system. These letters are composed out of sub-letters, called jamo, which correspond to individual phonemes, Jamo can, again, be decomposed into visual features that carry articulatory information about the phonemes, which is a characteristic of a featural writing system. The word 한글 (Hangul), for example, consists of two syllable blocks 한 (han), 글 (gul). The first block, for example, consists of three jamo, ㅎ (h), ㅏ (a), and ㄴ (n). As such, Hangul is a highly decomposable orthography, although there is little consensus about whether this decomposability has an effect on how readers

	k	n	w
Slots	1	-	-
ngram	500	3	-
Open ngram	500	3	3

Table 6: The optimal values for k and the various parameters in Korean.

baseline	**Slots**		**ngrams**		**Open ngrams**	
-	**Reg**	**Opt**	**Reg**	**Opt**	**Reg**	**Opt**
.197	.194	.228	.197	.240	.198	.238

Table 7: The optimal values for k and the various parameters in Korean.

of Korean behave. Rastle et al. (2019) found that readers of Hangul show no rigid transposition effects in a masked priming task, showing that Hangul is likely not processed like a purely alphabetic language. This still leaves the question of whether Korean shows a neighborhood effect, and, if we find such an effect, whether it behaves like an alphabetic language. We attempt to clarify this issue from a computational point of view, by applying the methods from the previous experiments to a large corpus of Korean Lexical Decision RT judgments on Hangul words (Yi et al., 2017).

5.1. Data and Preprocessing

We used the data of the Korean Lexicon Project (Yi et al., 2017), which consists of lexical decision judgments on 30,930 Korean words and non-words. As mentioned above, these words are written in syllable characters, or syllable blocks, which are built up out of a smaller set of phoneme letters, called jamo. Because there are many possible syllable blocks, e.g., the dataset we use contains 1391 unique blocks, we choose to decompose the syllable blocks into their constituent jamo, using the `jamo` Python package,[4] which led to a much smaller set of 66 jamo. For each word, we then simply concatenate the sets of jamo. As each syllable block can contain either two or three jamo[5], we chose to pad all blocks containing two jamo with a space character, because otherwise longer words would no longer be aligned. We also experimented with using the bare syllable blocks, and decomposing them into jamo without padding, both of which gave worse results.

5.2. Experiments

For the sake of convenience, we report on Experiments 1 and 2 simultaneously. As before, we optimized the parameters and the optimal k in a first step, the results of which are shown in Table 6.

The results of this analysis differ radically from the results on the alphabetic languages. First, for the alphabetic languages, a low number of neighbors worked well. Korean, instead seems to favor a really high number of neighbors for the ngram-based models, and a really low number of neighbors for the slot-based models. Note that the k value

[4]`https://github.com/JDongian/python-jamo`

[5]In non-computational terms, a syllable block can contain more than three jamo. In the unicode standard, however, some jamo characters are represented as complex characters, which we also adopted.

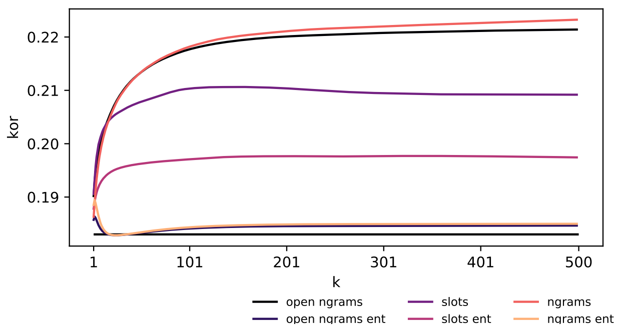

Figure 3: The R^2_{adj} over various values of k. As the figure shows, the entropy weighting did not have any effect. Note that the values of k for the Korean dataset are much higher.

reported for the ngram and open ngram models has an artificial plateau; we tested with values between 1 and 500 nearest neighbors, and found that the score of the ngram and open ngram models kept increasing, but very gradually so. The scores of the models on the test set are shown in Table 7. This shows another difference between datasets: the relative contribution of the baseline models seems to be relatively low, i.e., while logged frequency and length explains between .37 and .56 percent of variance in RT for the other language, they only explain .197 percent of variance in this case. The contribution of the non-optimized models is much smaller for Korean than for the other languages, implying that the neighborhood effect, while present, is expressed differently for Korean. Figure 3 shows the distribution over k for all Korean models. This shows the gradual upwards trend over increasing k, and also shows that feature weighting, both using H and J, has a negative effect for Korean. To confirm the trends, we again calculated 10,000 resampled bootstrap samples, which were compared in a pairwise fashion. This showed significant differences between all baselines and optimized models. Second, we also compared all optimized models to all weighted models, which showed that all weighted models were significantly worse than the optimized models. Weighting thus had the opposite effect for the Korean models, indicating another difference between Korean and the other languages. Figure 4 shows the distributon of the 10,000 bootstrapped resamples, showing that, in contrast to the other languages, the slot-based codes do not outperform other codes.

6. Conclusion

In conclusion, Table 8 shows the final regression coefficients for the best models. As shown, ngram models obtain the highest scores across all languages, indicating that this was an overlooked option in previous research. We demonstrate that it is possible to discover the neighborhood effect using a variety of feature sets, and that the feature sets which were in use before were probably not the best feature sets for discovering the feature sets. Although the difference between the highest score is relatively small, especially for Spanish and US English, we do obtain significantly better, and consistent, results across languages. This indicates that conceptualizing the orthographic neighborhood effect in terms of feature sets is a fruitful research di-

	Fra	Spa	Nld	Eng-UK	Eng-US	Kor
Features	ngram	ngram	ngram	ngram	ngram	ngram
Weighting	J	J	J	J	J	-
intercept	739.98	1060.055	585.06	650.67	762.18	632.56
freq	-39.36	-102.13	-29.89	-44.68	-47.32	-39.15
length	37.38	36.07	6.72	13.72	56.26	-21.47
rd	27.49	33.38	16.90	14.96	28.85	-19.18
R^2	.399	.605	.341	.363	.519	.223
ΔR^2	.015	.005	.022	.014	.007	.018
ΔR^2_{base}	.062	.041	.056	.025	.046	.040

Table 8: The final regression models. The top rows indicate the parameters of these models, the presence of weighting. The second part of the table shows their coefficients, while the bottom rows show their explained variance and change in explained variance.

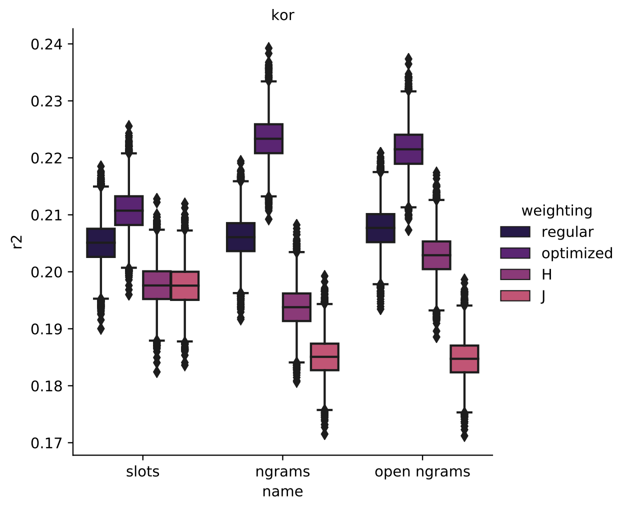

Figure 4: The R^2_{adj}, bootstrapped over 10,000 samples.

rection; perhaps other feature sets or other weighting methods are yet to be discovered. This also shows yet another contrast between Korean and other languages: while the neighborhood effect is a positive effect, it has a *negative* effect in Korean; words with more neighbors are read more slowly. This pattern has previously been observed in Chinese (Li et al., 2011; Wang et al., 2014; Perea, 2015; Chang et al., 2016), where the orthographic neighborhood, both when it is based on characters and strokes, has been observed to be negative. This shows that the neighborhood effect, although perhaps based on general cognitive principles such as co-activation of orthographically similar representations, can not be considered to be completely general effect. One possible confound, as noted by Perea (2015), is that the calculation of the neighborhood effect is highly dependent on the way words are conceptualized. For example, the concept of a word is less clear, and the task of word segmentation more difficult, for languages with syllabic or logographic writing systems, such as Japanese and Chinese. While the issue of word boundaries does not present itself in Hangul, the measurement of the neighborhood effect is highly dependent on whether the it is analyzed as a syllabary or as an alphabetic writing system. In future work, we would like to investigate this in more detail; one interesting direction to take this in would be to train representations of visual orthography directly learned from the visual modality, similar to the way the Triangle model (Harm and Seidenberg, 2004) was trained by Chang et al. (2019). From a computational point of view, it would be interesting to discover a less expensive way to calculate the neighborhood effect. Currently, the neighborhood effect relies on calculating the cosine distance between all words in the lexicon, which takes a lot of time and scales exponentially with the amount of words in the lexicon. In terms of general Natural Language Processing work, the discovery of new features for Hangul could perhaps be of use in training machine translation systems, as has been done for Chinese-Japanese-English machine translation (Zhang and Komachi, 2018).

7. Acknowledgments

We would like to thank Martina Pastore for preliminary work on Korean. The first author is supported by a PhD scholarship from the FWO Research Foundation - Flanders.

8. References

Aguasvivas, J., Carreiras, M., Brysbaert, M., Mandera, P., Keuleers, E., and Duñabeitia, J. A. (2018). Spalex: A spanish lexical decision database from a massive online data collection. *Frontiers in Psychology*, 9:2156.

Andrews, S. (1989). Frequency and neighborhood effects on lexical access: Activation or search? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(5):802.

Balota, D. A., Yap, M. J., Hutchison, K. A., Cortese, M. J., Kessler, B., Loftis, B., Neely, J. H., Nelson, D. L., Simpson, G. B., and Treiman, R. (2007). The english lexicon project. *Behavior research methods*, 39(3):445–459.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62.

Brillouin, L. (1953). The negentropy principle of information. *Journal of Applied Physics*, 24(9):1152–1163.

Brysbaert, M. and New, B. (2009). Moving beyond kučera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41(4):977–990.

Chang, Y.-N., Welbourne, S., and Lee, C.-Y. (2016). Exploring orthographic neighborhood size effects in a computational model of chinese character naming. *Cognitive psychology*, 91:1–23.

Chang, Y.-N., Furber, S., Ralph, M. L., and Welbourne, S. (2019). A computational model of normal and impaired lexical decision: Graded semantic effects. *BioRxiv*, page 708156.

Coltheart, M. (1977). Access to the internal lexicon. *The psychology of reading*.

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

Daelemans, W., Van Den Bosch, A., and Weijters, T. (1997). Igtree: using trees for compression and classification in lazy learning algorithms. In *Lazy learning*, pages 407–423. Springer.

Daelemans, W., Zavrel, J., Van der Sloot, K., and Van den Bosch, A., (1998). *TiMBL: Tilburg Memory-Based Learner-version 1.0-Reference Guide*.

Davis, C. J. and Bowers, J. S. (2006). Contrasting five different theories of letter position coding: Evidence from orthographic similarity effects. *Journal of Experimental Psychology: Human Perception and Performance*, 32(3):535.

Davis, C. J. (2010). The spatial coding model of visual word identification. *Psychological Review*, 117(3):713.

Dehaene, S., Cohen, L., Sigman, M., and Vinckier, F. (2005). The neural code for written words: a proposal. *Trends in cognitive sciences*, 9(7):335–341.

Dehaene, S. (2009). *Reading in the brain: The new science of how we read*. Penguin.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Ferrand, L., New, B., Brysbaert, M., Keuleers, E., Bonin, P., Méot, A., Augustinova, M., and Pallier, C. (2010). The french lexicon project: Lexical decision data for 38,840 french words and 38,840 pseudowords. *Behavior Research Methods*, 42(2):488–496.

Frost, R. (2012). A universal approach to modeling visual word recognition and reading: Not only possible, but also inevitable. *Behavioral and brain sciences*, 35(5):310–329.

Grainger, J. and Van Heuven, W. J. (2004). Modeling letter position coding in printed word perception. In Patrick Bonin, editor, *Mental lexicon: "Some words to talk about words"*, page 1–23. Nova Science Publishers.

Grainger, J. (1990). Word frequency and neighborhood frequency effects in lexical decision and naming. *Journal of memory and language*, 29(2):228–244.

Grainger, J. (2008). Cracking the orthographic code: An introduction. *Language and cognitive processes*, 23(1):1–35.

Grainger, J. (2018). Orthographic processing: A "mid-level" vision of reading. *The Quarterly Journal of Experimental Psychology*, 71(2):335–359.

Harm, M. W. and Seidenberg, M. S. (2004). Computing the meanings of words in reading: cooperative division of labor between visual and phonological processes. *Psychological review*, 111(3):662.

Keuleers, E., Brysbaert, M., and New, B. (2010a). Subtlex-nl: A new measure for dutch word frequency based on film subtitles. *Behavior research methods*, 42(3):643–650.

Keuleers, E., Diependaele, K., and Brysbaert, M. (2010b). 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*, 1:174.

Keuleers, E., Lacey, P., Rastle, K., and Brysbaert, M. (2012). The british lexicon project: Lexical decision data for 28,730 monosyllabic and disyllabic english words. *Behavior research methods*, 44(1):287–304.

Kinoshita, S. and Norris, D. (2013). Letter order is not coded by open bigrams. *Journal of memory and language*, 69(2):135–150.

Lad, F., Sanfilippo, G., Agro, G., et al. (2015). Extropy: complementary dual of entropy. *Statistical Science*, 30(1):40–58.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Li, Q.-L., Bi, H.-Y., Wei, T.-Q., and Chen, B.-G. (2011). Orthographic neighborhood size effect in chinese character naming: Orthographic and phonological activations. *Acta psychologica*, 136(1):35–41.

McClelland, J. L. and Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological review*, 88(5):375.

Miikkulainen, R. (1993). *Subsymbolic natural language processing: An integrated model of scripts, lexicon, and memory*. MIT Press.

Miikkulainen, R. (1997). Dyslexic and category-specific aphasic impairments in a self-organizing feature map model of the lexicon. *Brain and language*, 59(2):334–366.

New, B., Pallier, C., Ferrand, L., and Matos, R. (2001). Une base de données lexicales du français contemporain sur internet: Lexique^{TM}//a lexical database for contemporary french: Lexique^{TM}. *L'année psychologique*, 101(3):447–462.

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

Pae, H. K. (2011). Is korean a syllabic alphabet or an alphabetic syllabary. *Writing Systems Research*, 3(2):103–115.

Perea, M. and Lupker, S. J. (2004). Can caniso activate casino? transposed-letter similarity effects with nonadjacent letter positions. *Journal of memory and language*, 51(2):231–246.

Perea, M., Duñabeitia, J. A., and Carreiras, M. (2008). Transposed-letter priming effects for close versus distant transpositions. *Experimental Psychology*, 55(6):384–393.

Perea, M., Abu Mallouh, R., and Carreiras, M. (2010).

The search for an input-coding scheme: Transposed-letter priming in arabic. *Psychonomic Bulletin & Review*, 17(3):375–380.

Perea, M. (2015). Neighborhood effects in visual word recognition and reading. *The Oxford Handbook of Reading*, page 76.

Rastle, K., Lally, C., and Lee, C. H. (2019). No flexibility in letter position coding in korean. *Journal of Experimental Psychology: Human Perception and Performance*, 45(4):458.

Rumelhart, D. E. and Siple, P. (1974). Process of recognizing tachistoscopically presented words. *Psychological review*, 81(2):99.

Sampson, G. (1985). *Writing systems*. London, U.K.: Hutchinson.

Schoonbaert, S. and Grainger, J. (2004). Letter position coding in printed word perception: Effects of repeated and transposed letters. *Language and Cognitive Processes*, 19(3):333–367.

Schrödinger, E. (1944). *What is Life – the Physical Aspect of the Living Cell*. Cambridge University Press.

Stanfill, C. and Waltz, D. L. (1986). Toward memory-based reasoning. *Commun. ACM*, 29(12):1213–1228.

Tulkens, S., Sandra, D., and Daelemans, W. (2018a). From strings to other things: Linking the neighborhood and transposition effects in word reading. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 75–85.

Tulkens, S., Sandra, D., and Daelemans, W. (2018b). Wordkit: a python package for orthographic and phonological featurization. In Nicoletta Calzolari (Conference chair), et al., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA).

Van Assche, E. and Grainger, J. (2006). A study of relative-position priming with superset primes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 32(2):399.

Van Heuven, W. J., Mandera, P., Keuleers, E., and Brysbaert, M. (2014). Subtlex-uk: A new and improved word frequency database for british english. *Quarterly Journal of Experimental Psychology*, 67(6):1176–1190.

Velan, H. and Frost, R. (2007). Cambridge university versus hebrew university: The impact of letter transposition on reading english and hebrew. *Psychonomic Bulletin & Review*, 14(5):913–918.

Velan, H. and Frost, R. (2009). transposition effects are not universal: The impact of transposing letters in hebrew. *Journal of Memory and Language*, 61(3):285–302.

Velan, H., Deutsch, A., and Frost, R. (2013). The flexibility of letter-position flexibility: Evidence from eye movements in reading hebrew. *Journal of Experimental Psychology: Human Perception and Performance*, 39(4):1143.

Wang, J., Tian, J., Han, W., Liversedge, S. P., and Paterson, K. B. (2014). Inhibitory stroke neighbour priming in character recognition and reading in chinese. *The Quarterly Journal of Experimental Psychology*, 67(11):2149–2171.

Whitney, C. and Cornelissen, P. (2008). Seriol reading. *Language and Cognitive Processes*, 23(1):143–164.

Whitney, C. (2001). How the brain encodes the order of letters in a printed word: The seriol model and selective literature review. *Psychonomic Bulletin & Review*, 8(2):221–243.

Wickelgren, W. A. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review*, 76(1):1.

Yarkoni, T., Balota, D., and Yap, M. (2008). Moving beyond coltheart's n: A new measure of orthographic similarity. *Psychonomic Bulletin & Review*, 15(5):971–979.

Yi, K., Koo, M., Nam, K., Park, K., Park, T. ., Bae, S., Lee, C. H., Lee, H.-W., and Cho, J.-R. (2017). The korean lexicon project: A lexical decision study on 30,930 korean words and nonwords. *The Korean Journal of Cognitive and Biological Psychology*, pages 395–410.

Zhang, L. and Komachi, M. (2018). Neural machine translation of logographic languages using sub-character level information. *arXiv preprint arXiv:1809.02694*.