# How (Non-)Optimal is the Lexicon?

**Tiago Pimentel**[*,ð] **Irene Nikkarinen**[*,ð,ɤ] **Kyle Mahowald**[ŋ]
**Ryan Cotterell**[ð,ʕ] **Damián E. Blasi**[ɘ,ɘ,ɒ]
[ð]University of Cambridge [ɤ]Yle [ŋ]University of California, Santa Barbara [ʕ]ETH Zürich
[ɘ]Harvard University [ɘ]MPI for Evolutionary Anthropology [ɒ]HSE University
tp472@cam.ac.uk, irene.nikkarinen@gmail.com, mahowald@ucsb.edu
ryan.cotterell@inf.ethz.ch, dblasi@fas.harvard.edu

## Abstract

The mapping of lexical meanings to wordforms is a major feature of natural languages. While usage pressures might assign short words to frequent meanings (Zipf's law of abbreviation), the need for a productive and open-ended vocabulary, local constraints on sequences of symbols, and various other factors all shape the lexicons of the world's languages. Despite their importance in shaping lexical structure, the relative contributions of these factors have not been fully quantified. Taking a coding-theoretic view of the lexicon and making use of a novel generative statistical model, we define upper bounds for the compressibility of the lexicon under various constraints. Examining corpora from 7 typologically diverse languages, we use those upper bounds to quantify the lexicon's optimality and to explore the relative costs of major constraints on natural codes. We find that (compositional) morphology and graphotactics can sufficiently account for most of the complexity of natural codes—as measured by code length.

## 1 Introduction

Communication through language can be modeled under Shannon's classic communication framework (Shannon, 1948). Under this perspective, linguistic utterances are **codes**—which need to be decoded by a **receiver** (listener) who is interested in the **message** (meaning) they encode. Famously, Zipf (1949) posited that language users shape these codes so to accommodate the **principle of least effort**. The most widely discussed and investigated empirical evidence for this feature is the so-called **law of abbreviation**, an ostensive negative correlation between word frequency and word length (Zipf, 1935; Bentz and Ferrer-i-Cancho, 2016). Communication effort decreases by encoding frequent messages in shorter words.

[*]Equal contribution

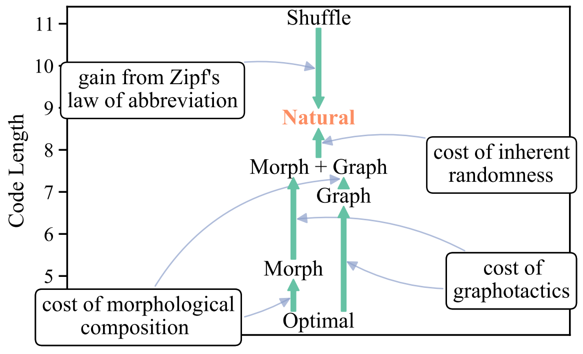

Figure 1: The average code length—under our coding schemes—on a representative language (Finnish). The distance between the baselines can be thought of as the cost of each constraint added to the system.

This correlation, however, is characteristically modest. There are many instances of short low-frequency words, like *wen* and *jib* in English,[1] and long frequent words, like *happiness* and *anything*. While the lexicon might be shaped by economy of expression, it is clearly not fully optimized for it. There are multiple—possibly competing—reasons why this could be the case.

First of all, the sequence of speech sounds, signs, or orthographic characters that serve as building blocks in a language comply with specific rules. These are referred to as **phonotactics** (in the case of speech sounds) and **graphotactics** (in written language).[2] On top of these constraints, the lexicons of many languages of the world re-use subparts of words; these sub-parts can be productively composed to produce new meanings—which is referred to as **morphological composition**. This largely determines the family of wordforms associated with a given basic meaning—for instance, given the wordform *health* and its meaning, the nominal morphology of English readily provides

[1]These mean, respectively, a benign tumor on the skin and a triangular sail on a boat.

[2]All languages impose these constraints on their wordforms, which might be leveraged for production and learnability (Vitevitch and Luce, 1999; Boersma, 1998).

the forms for many of its derived meanings, including *healthy*, *unhealthy*, *healthier*, etc.

Beyond these well-attested constraints, it might be argued that the negative correlation between the length of a word and its frequency is not the locus of optimization given the economy of expression pressure. Instead, wordforms might be efficiently encoding meanings based on their contextual surprise rather than frequency (Piantadosi et al., 2011). Finally, there is no reason to expect lexicons to be *fully optimized* for the economy of expression—this factor might steer languages in a given direction, but there is certainly room for non-compliance. Languages are, after all, not engineered systems but cultural artifacts.

In this paper, we examine how *marginally non-optimal* the lexicon is by taking the vantage point of the law of abbreviation. We develop a method to quantify the role of different linguistic constraints on determining wordforms, and we produce estimates on how compressible the lexicon could be in their absence (including morphology and phonotactics/graphotactics). We thus define an upper bound for the compressibility of a lexicon optimized purely for word length efficiency.

## 2 (Non-)Optimality in the Lexicon

As stated above, our notion of optimality is derived from Zipf's principle of least effort in the form of the law of abbreviation (Zipf, 1949; Mandelbrot, 1953; Ferrer-i-Cancho et al., 2020). However, this is by no means the only theory under which wordforms are optimized for encoding their messages.

One influential hypothesis is that languages optimize for uniform information density (Fenk and Fenk, 1980; Aylett and Turk, 2004; Levy and Jaeger, 2007)—roughly keeping the amount of information conveyed in a unit of time constant. In an information-theoretic setting, this would be equivalent to maximizing the use of a noisy channel between the speaker and an audience—keeping the transmission rate close to the channel capacity.

Under this view, it is not necessarily the case that words should be as short as possible. Rather, words that are infrequent or typically less predictable in context should be longer and take more time to produce—perhaps because the increased duration makes them more robust to noise. Consistent with this perspective, it has been shown that, in production, words with higher information content take longer to pronounce (Bell et al., 2003; Jurafsky et al., 2001; Gahl, 2008). Additionally, words which are typically predictable in context are shorter than words which are less predictable in context (Piantadosi et al., 2011).

On another note, a purely coding-theoretically efficient language could make the lexical codes context dependent (Piantadosi et al., 2012), since context often disambiguates words (Dautriche et al., 2018; Pimentel et al., 2020a). Additionally, the meaning or message being conveyed by a given word might bias its form. Within languages, there seems to be a pressure for more semantically similar words to also be more phonologically similar (Monaghan et al., 2014; Dingemanse et al., 2015; Dautriche et al., 2017; Pimentel et al., 2019). Across languages, words for the same referents exhibit detectable patterns in term of their phonological makeup (Blasi et al., 2016), phonotactics (Pimentel et al., 2021b), as well as word length (Lewis and Frank, 2016)—this is driven by semantic features such as size, quality or complexity. Finally, there is a cross-linguistic tendency for lexicons to place higher surprisal in word-initial segments (van Son and Pols, 2003a,b; King and Wedel, 2020; Pimentel et al., 2021a) making words more constrained in their choice of final segments. These aspects of language might also collide with a purely Zipfian conception of lexicon optimality.

In this work, however, we consider optimality exclusively in the Zipfian sense of compressibility, and we ask how far natural language lexicons are from accommodating to this paradigm. We build a number of models that differ in relation to whether they accommodate to the law of abbreviation, to compositional morphology and to graphotactics. The comparison among these systems allows us to explore the extent to which each part of the linguistic system contributes to the overall cost of the linguistic code. It should be noted, though, that the consequences of unmodeled sources of structure in the lexicon (such as persistent sound-meaning associations or the adaptation of the code to surprisal effects) will forcibly be confounded with the overall lack of fit between our models and the data.

The **morphological cost** —i.e. the cost of morphology to a code's length—is associated with the fact that, across many languages, words are often constructed of meaningful sub-parts that are productively reused across the lexicon. Practically, this means that the wordforms of different meanings might not be independent if they overlap in a particular dimension that is captured by the morphology of the language. For instance, most wordforms that express two or more referents of a kind share a word-final suffix *-s* in English (*towers*, *cats*, *ideas*, etc). We treat this cost by considering optimal codes where the basic unit in the lexicon is not the word but sub-pieces, as determined by the unsupervised morphological parser Morfessor (Creutz and Lagus, 2007; Smit et al., 2014).[3] Under this regime, a word like *unmatched* is parsed into the tokens *un*, *match*, and *ed*.

The **graphotactic cost** concurrently imposes a set of additional constraints, determining which sequences of grapheme segments can constitute a valid wordform in a given language. While the main driver of these lexical constraints is actually phonotactics—which imposes rules dictating the possible *phoneme* sequences—we focus on graphotactics because our object of study is written language corpora. The degree to which phonotactics and graphotactics mirror each other vary substantially across languages; thus, in this work (which uses corpora from Wikipedia) we make our claims about language in the written modality and leave it to future work to generalize this work to the phonological domain. This could be done by applying the same method to phonemic representations of words.

## 3 A Coding-theoretic View of the Lexicon

This paper treats the **lexicon**, which we define as a set of pairs: $\mathcal{L} = \{(m_n, \boldsymbol{w}_n)\}_{n=1}^{N}$. In general, this set will be infinite; $m_n$ refers to a **lexical meaning**, taken from an abstract set $\mathcal{M}$, and $\boldsymbol{w}_n$ refers to a **wordform**, taken from $\Sigma^*$, the Kleene closure of a grapheme alphabet $\Sigma$.[4] When the exact index is unnecessary in context, we will drop the subscripted $n$; and we make use of uppercase letters to refer to random variables (e.g. $M$ or $W$) where necessary. We will write meanings in typewriter font, e.g. `cat`, and wordforms in italics: *cat* (English), *kissa* (Finnish).

Viewing the lexicon from a coding-theoretic perspective, we consider the mapping from meaning to form as a **code**: $\mathbb{C} : \mathcal{M} \to \Sigma^*$. Every language comes endowed with a **natural code** $\mathbb{C}_{\textit{nat}}$, which is the observed mapping from lexical meanings to forms. As an example, consider the meaning `cat` and its Finnish form: we have $\mathbb{C}_{\textit{nat}}(\texttt{cat}) = \textit{kissa}$. The topic of interest in this paper is the efficiency of language's natural codes.

**The space of meanings and lexical ambiguity.** The space of meanings $\mathcal{M}$ is non-trivial to define, but could be operationalized as $\mathbb{R}^d$, which is infinite, continuous and uncountable (Pilehvar and Camacho-Collados, 2020). Meanwhile, the space of wordforms $\Sigma^*$ is also infinite, but discrete and countable. As such, many meanings $m_n$ must be mapped to the same form, resulting in lexical ambiguity. See Pimentel et al. 2020a for a longer discussion on these operationalizations. In this work, though, we do not engage with such ambiguity, considering $\mathcal{M}$ as an abstract set of meanings, each of which defined by a distinct wordform—i.e. the code $\mathbb{C}_{\textit{nat}}$ is a bijection. A consequence of this strategy is that we take the space of meanings to be infinite, but discrete and countable; we only distinguish as many meanings as there are words, therefore, we end up with a countable number of meanings. Additionally, by considering a distinct meaning $m_n$ for each wordform $\boldsymbol{w}_n$ in the lexicon, we only consider codes with as much lexical ambiguity as in the original language.[5]

### 3.1 Words as Meanings

The **unigram distribution** represents the frequency of each wordform in a text, i.e. the probability of a token without conditioning on context $p(W = \textit{kissa})$. In this work, though, we assume the unigram distribution is a distribution over $\mathcal{M}$, e.g. $p(M = \texttt{cat})$—this way we can analyze how changing the code $\mathbb{C}$ would affect its efficiency.

As stated above, though, we take $\mathbb{C}_{\textit{nat}}$ to be a bijection. Such an assumption implies there is a deterministic function from wordforms to meanings in a specific lexicon $\mathbb{C}_{\textit{nat}}^{-1}(\boldsymbol{w}) = m$. Probabilistically speaking, we write

$$p(M = m \mid W = \boldsymbol{w}) = \mathbb{1}\Big\{m = \mathbb{C}_{\textit{nat}}^{-1}(\boldsymbol{w})\Big\} \quad (1)$$

$$p(W = \boldsymbol{w} \mid M = m\} = \mathbb{1}\Big\{\boldsymbol{w} = \mathbb{C}_{\textit{nat}}(m)\Big\} \quad (2)$$

[3]We also present results using the additional sub-word tokenizers: byte pair encoding (Gage, 1994; Sennrich et al., 2016) and word piece (Schuster and Nakajima, 2012). See Bostrom and Durrett (2020) for a discussion of the tradeoffs of these schemes, in terms of performance and compressibility.

[4]This alphabet is augmented with an end-of-word symbol.

[5]Lexical ambiguity allows the mapping of multiple meanings to the same wordform and, in doing so, it enables the preferential re-use of short words (Piantadosi et al., 2012). Thus, the mapping of multiple meanings to the same form could be a source of efficiency in the lexicon (Fenk-Oczlon and Fenk, 2008; Ferrer-i-Cancho and Vitevitch, 2018; Casas et al., 2019; Trott and Bergen, 2020; Xu et al., 2020). Nonetheless, we do not treat it explicitly here.

This mapping implies

$$p(M = m_n) = \sum_{\boldsymbol{w} \in \Sigma^*} p(M = m_n, W = \boldsymbol{w}) \quad (3)$$
$$= p(W = \boldsymbol{w}_n)$$

Given this equality, we can reduce the problem of estimating the unigram distribution over meanings $p(m)$ to the one over wordforms $p(\boldsymbol{w})$.

### 3.2 Code-length and optimality

As stated above, we assume the unigram distribution to be a distribution over $\mathcal{M}$. We now define the **cost** of a code as its expected length:

$$\mathrm{cost}(\mathbb{C}) = \sum_{m \in \mathcal{M}} p(m)\, |\mathbb{C}(m)| \quad (4)$$

A smaller cost, then, implies a more **efficient** code. The famous **source-coding theorem** of Shannon (1948) gives us a theoretical limit on coding cost:

$$\mathrm{H}(M) \leq \mathrm{cost}(\mathbb{C}_\star) < \mathrm{H}(M) + 1 \quad (5)$$

where we define $\mathbb{C}_\star$ to be the most efficient code, and where $\mathrm{H}(M)$ is the entropy of distribution $p$:

$$\mathrm{H}(M) = \sum_{m \in \mathcal{M}} p(m) \, \log_{|\Sigma|} \frac{1}{p(m)} \quad (6)$$

According to the source-coding theorem, if we know the true distribution $p$ over lexical meanings, then we know how to optimally code them. This turns the problem of estimating the efficiency of the lexicon into the one of estimating the entropy of an unknown discrete distribution $p$, a well-defined task with a pool of previous work (Miller, 1955; Antos and Kontoyiannis, 2001; Paninski, 2003; Archer et al., 2014). Because the distributions over wordforms and meanings are equivalent, we estimate the entropy $\mathrm{H}(M)$ using wordforms:

$$\mathrm{H}(M) = \mathrm{H}(W) = \sum_{\boldsymbol{w} \in \Sigma^*} p(\boldsymbol{w}) \, \log_{|\Sigma|} \frac{1}{p(\boldsymbol{w})} \quad (7)$$

### 3.3 Finite and Infinite Support

This section reviews a few technical results as regards the construction of codes from a probability distribution. If $p$ had finite support—i.e. there were a finite set of possible meanings or wordforms—a simple Huffman encoding (Huffman, 1952) would give us an optimal code for our lexicon. However, this is not the case—$p(\boldsymbol{w})$ has support on all of $\Sigma^*$—so we might need a more complex strategy to get such a code. Linder et al. (1997) proved the existence of an optimal encoding for a distribution with infinite support, given that it has finite entropy.

**Proposition 1.** *If distribution $p(\boldsymbol{w})$ has finite entropy, i.e. $\mathrm{H}(W) < \infty$, then there exists an optimal encoding for it such that:* $\mathrm{cost}(\mathbb{C}_\star) < \mathrm{H}(M) + 1$.

*Proof.* See Linder et al. (1997). □

Luckily, under a weak assumption, this is the case for a well-trained language model.

**Definition 1.** *Language model $p(\boldsymbol{w})$ is $\varepsilon$-smooth if for all histories $\boldsymbol{h} \in \Sigma^*$ we have $p(\texttt{EoW} \mid \boldsymbol{h}) \geq \varepsilon$.*[6]

This fairly weak assumption states that partial wordforms have a lowerbound on their probability of ending. As such, there is an upperbound on the probability of a wordform which decreases exponentially with its length. Armed with this assumption, we can now show that any $\varepsilon$-smooth language model has a finite entropy.

**Proposition 2.** *If a language model $p(\boldsymbol{w})$ is $\varepsilon$-smooth, then its entropy is finite, i.e.* $\mathrm{H}(W) < \infty$.

*Proof.* See App. C. □

Safe-guarded by Propositions 1 and 2, we now train a model to capture the unigram distribution. We will then use this model to estimate the code-length of an optimal lexicon.

## 4 Modeling the Unigram Distribution and its Challenges

Zipf's (1935) law states that the frequency of a word in a corpus is inversely proportional to its rank, resulting in a power-law distribution where a small subset of the words dominate the corpus. As such, naïvely training a character-level model on a language's tokens (i.e. predicting non-contextual wordforms with their natural corpus frequencies) would be unlikely to capture morphological regularities (Goldwater et al., 2011). Furthermore, it would burden the model to learn a mostly arbitrary assignment between form and frequency. As an example, the English verb *make* is much more common than the nouns *cake* and *lake*, even if graphotactically they may be equally probable.

A closer inspection of English shows that most frequent words tend to come from closed lexical classes including articles, pronouns, prepositions, and auxiliaries, such as *the*, *of*, *it* and *be* (Sinclair, 1999). These words tend to be short and manifest fossilized graphotactics (and phonotactics) as well

[6]Under this assumption our language model is also consistent, as defined by Welleck et al. (2020)—sequences with infinite length have asymptotically zero probability mass.

as a more abundant prevalence of otherwise rare segments, such as the voiced and voiceless dental fricatives (orthographically expressed with *th*). These rare segments would be overrepresented in such a naïve training regime, making it hard for the character-level model to correctly represent the language's graphotactics.

In order to address the problem of skewed frequencies, we use a novel neuralization of Goldwater et al.'s (2011) two-stage model to capture the unigram distribution. This model consists of two components: a wordform **generator** and a token frequency **adaptor**. The generator is a character-level model which produces wordforms, for which we use an LSTM;[7] this model should place similar probability mass on graphotactically "good" wordforms, such as *make*, *cake*, and *lake*. Meanwhile, the adaptor sets the frequency with which these wordforms will appear as tokens. Following Goldwater et al., we base our adaptor on the Pitman–Yor Chinese restaurant process (PYCRP; Pitman and Yor, 1997), which allows the adaptor to model a power-law distribution; this model is then responsible for capturing the fact that *make* is a more frequent token than *cake*, and *lake*.

## 4.1 A Two-stage Model

The generative process of our two-stage model is presented graphically in Fig. 2. Our generator is a character-level LSTM language model, which generates a potentially infinite number of i.i.d. wordforms $\{\boldsymbol{\ell}_k\}_{k=1}^{K}$. Independently, the PYCRP adaptor assigns each observed token in a dataset to a cluster $\{z_n\}_{n=1}^{N}$. In the literature, the value of $z_n$ is the "table assignment" of the nth token. These clusters are then used as lookup indices to the wordforms, producing the observed word tokens $\{\boldsymbol{w}_n\}_{n=1}^{N}$ where $\boldsymbol{w}_n = \boldsymbol{\ell}_{z_n}$. In general $N \gg K$, so tokens with the same wordform are grouped in few clusters. In this way, the adaptor sets the frequency with which wordforms appear as tokens in a corpus by defining each cluster's probability.

**Generating Wordforms.** As mentioned above, wordforms are sampled i.i.d. from a distribution $p_\phi$ over strings defined by the generator. Specifically, this distribution over forms is defined as follows:

$$p_\phi(\boldsymbol{\ell}) = \prod_{t=1}^{|\boldsymbol{\ell}|} p_\phi(\ell_t \mid \boldsymbol{\ell}_{<t}) \tag{8}$$

[7]LSTMs have been shown to be able to model phonotactics well by Pimentel et al. (2020b), and so we expect them to also work well with graphotactics.

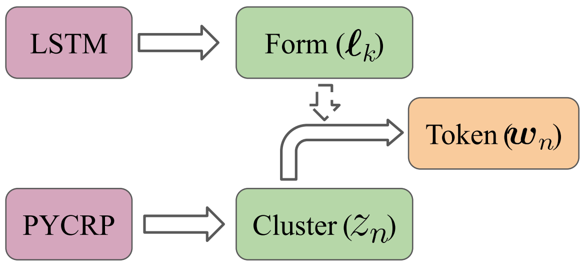

Figure 2: A diagram of the two-stage model. The LSTM generates wordforms ($\boldsymbol{\ell}_k$). The PYCRP samples cluster assignments ($z_n$). Cluster assignments are then used to lookup a form for each token ($\boldsymbol{w}_n = \boldsymbol{\ell}_{z_n}$). In this Figure, models are in **magenta**, latent variables in **green** and observed variable in **orange**.

where $\boldsymbol{\ell}$ is a vector of characters forming a word and $\ell_t$ is its $t^{\text{th}}$ character.[8] Each of these characters is encoded with a lookup vector, producing representations $\mathbf{e}_t \in \mathbb{R}^{d_1}$ where $d_1$ is the embedding size. These embeddings are then used as input to an LSTM (Hochreiter and Schmidhuber, 1997), producing the representations $\mathbf{h}_t \in \mathbb{R}^{d_2}$, where $d_2$ is the size of the LSTM's hidden layer. The LSTM output is further used to obtain the distribution over potential characters:

$$p_\phi(\ell_t \mid \boldsymbol{\ell}_{<t}) = \mathrm{softmax}(\mathbf{W}\,\mathbf{h}_t + \mathbf{b}) \tag{9}$$

In this equation, both $\mathbf{W} \in \mathbb{R}^{|\Sigma| \times d_2}$ and $\mathbf{b} \in \mathbb{R}^{|\Sigma|}$ are learnable parameters and the zero vector is used as the initial hidden state $\mathbf{h}_0$. The distribution $p_\phi$, representing the generator, is then used to generate the set of wordforms $\{\boldsymbol{\ell}_k\}_{k=1}^{K}$, which is expected to represent the graphotactics and morphology of the language. Notedly, these wordforms do not explicitly capture any notion of token frequency.[9]

**Adapting Word Frequencies.** The adaptor is responsible for modeling the word frequencies, and it has no explicit notion of the wordforms themselves. The PYCRP assigns each token $n$ to a cluster $z_n$. Each cluster $z_n$, in turn, has an associated wordform $\boldsymbol{\ell}_{z_n}$, sampled from the generator. Consequently, all instances in a cluster share the same wordform. The probability of an instance $n$ being assigned to cluster $z_n$ is defined as follows:

$$p(Z_n = z_n \mid \mathbf{z}_{<n}) \tag{10}$$
$$\propto \begin{cases} c_{<n}^{(z_n)} - a & 1 \le z_n \le K_{<n} \ \textit{(old cluster)} \\ a \cdot K_{<n} + b & z_n = K_{<n} + 1 \ \textit{(new cluster)} \end{cases}$$

[8]We note two subscripts are used here: $k$ refers to the $k^{\text{th}}$ wordform, while $t$ indexes the $t^{\text{th}}$ character in the wordform.

[9]This generative process allows the same wordform to be sampled multiple times, as they are generated i.i.d.

In this equation, $K_{<n}$ is the current number of populated clusters; while $\mathrm{c}_{<n}^{(z_n)}$ is the number of instances currently assigned to cluster $z_n$. The PYCRP has two hyperparameters: $0 \leq a < 1$ and $b \geq 0$. The parameter $a$ controls the rate in which the clusters grow (Teh, 2006), while $b$ controls an initial preference for dispersion. Together, these ensure the formation of a long-tail—concocting a power-law distribution for the cluster frequencies. This property allows a cluster with wordform *make*, for example, to have an exponentially larger frequency than its graphotactic neighbor *cake*.

**Modeling Word Tokens.** Finally, given the set of wordforms and the cluster assignments, defining the form associated with a token is deterministic. Since each cluster only contains instances of one wordform, the form of a token is defined looking up the label of the cluster it was assigned to $\boldsymbol{\ell}_{z_n}$:

$$p(W_n = \boldsymbol{w}_n \mid z_n, \boldsymbol{\ell}) = \mathbb{1}\{\boldsymbol{w}_n = \boldsymbol{\ell}_{z_n}\} \quad (11)$$

This way, the adaptor captures the frequency information of the words in the corpus—whereas the generator can focus on learning the language's graphotactics and morphology.

**Model training.** Unfortunately, we cannot directly infer the parameters of our model with a closed form solution. We thus use a solution akin to expectation maximization (Wei and Tanner, 1990): We freeze our LSTM generator while learning the PYCRP parameters, and *vice versa*. The PYCRP is trained using Gibbs sampling. For each token, we fix all cluster assignments $\mathbf{z}_{-n}$ except for one $z_n$. This cluster is then re-sampled from the marginal $p(Z_n = z_n \mid \mathbf{z}_{-n}, \boldsymbol{\ell}, \boldsymbol{w}_n)$, where we have access to $\boldsymbol{w}_n$ since it is an observed variable. During this optimization small clusters may vanish, and new clusters $z_n = K+1$ (previously with no samples) may be created. This procedure, thus, may also produce new sets of wordforms $\{\boldsymbol{\ell}_k\}_{k=1}^{K'}$, composed of the populated clusters' labels (where $K'$ is the new number of clusters). We assume the distribution of these wordforms—which have dampened frequencies—to be more balanced than in the original full set of word tokens. The LSTM is trained using stochastic gradient descent, minimizing the cross-entropy of precisely this set of cluster's wordforms. As such, it is expected to be a more representative model of a language's graphotactics; the irregular common words are less dominant in this training set. We give a longer explanation of our model training procedure, together with the used hyperparameters, in App. B.

### 4.2 A More Intuitive Explanation

Despite its slightly odd formulation, the two-stage model has an intuitive interpretation. Once we have learned (and fixed) its parameters, we obtain the marginal probability of a wordform as:

$$p(\boldsymbol{w}) = \quad (12)$$
$$\underbrace{\frac{c_{\boldsymbol{w}} - \overbrace{n_{\boldsymbol{w}} \cdot a}^{\text{smoothing factor}}}{|\mathbf{z}| + b}}_{\text{smoothed unigram frequencies}} + \underbrace{\frac{(a \cdot K + b)}{|\mathbf{z}| + b}}_{\text{interpolation weight}} \cdot \underbrace{p_\phi(\boldsymbol{w})}_{\text{LSTM}}$$

In this equation, $c_{\boldsymbol{w}}$ is the count of tokens with form $\boldsymbol{w}$ in the training set, while $n_{\boldsymbol{w}}$ is the number of distinct clusters with this same form. The model interpolates between a smoothed unigram corpus frequency and the probability an LSTM gives the analyzed wordform. This interpolation enables the model to place a non-zero probability mass on all possible wordforms—thus modeling an open vocabulary and having infinite support—while also placing a large probability mass on frequent wordforms. Furthermore, the smoothing factors per word type, together with the interpolation weight, are holistically learned by the PYCRP model using the training set.[10]

## 5 Experimental Setup

### 5.1 Evaluation

The value in which we are interested in this work is the expected cost of a code, given in eq. (4). We can easily estimate this value for a natural code by using its sample estimate:

$$\mathrm{cost}(\mathbb{C}_{nat}) \approx \frac{1}{N}\sum_{n=1}^{N} |\mathbb{C}_{nat}(m_n)| = \frac{1}{N}\sum_{n=1}^{N} |\boldsymbol{w}_n| \quad (13)$$

For an optimal code, we can upperbound it using the entropy of the distribution, while the entropy itself can be upperbounded by the cross-entropy of a model on it. We can compute this upperbound with a sample estimate of the cross-entropy:

$$\mathrm{cost}(\mathbb{C}_\star) \leq \mathrm{H}(W) + 1 \leq \mathrm{H}_\theta(W) + 1 \quad (14)$$
$$\lesssim \frac{1}{N}\sum_{n=1}^{N} \log_{|\Sigma|} \frac{1}{p_\theta(\boldsymbol{w}_n)} + 1$$

[10]Our model consistently produced lower cross-entropies (on held out tokens) to the ones of an LSTM baseline naïvely trained on a language's tokens.

In practice, we get a tighter estimate by using the Shannon (1948) code's lengths directly:

$$\mathrm{cost}(\mathbb{C}_\star) \lesssim \frac{1}{N}\sum_{n=1}^{N}\left\lceil \log_{|\Sigma|}\frac{1}{p_\theta(\boldsymbol{w}_n)}\right\rceil \tag{15}$$

where $\lceil\cdot\rceil$ is the ceiling operation.

### 5.2 Morphological Constraints

As mentioned in §2, we use Morfessor (Smit et al., 2014) to tokenize our corpus into morphological units. Morfessor is a method for finding morphological segmentations from raw text data. As an unsupervised model, Morfessor is inherently noisy, but we take it as a proxy for a language's morphological segmentation. To compare the robustness of our results across different unsupervised segmentation algorithms, though, we also run our experiments using byte pair encoding (BPE; Gage, 1994; Sennrich et al., 2016) and WordPieces (Schuster and Nakajima, 2012).

We train Morfessor on all pre-tokenized sentences in our language-specific Wikipedia corpus (described in §5.4). With this pre-trained model in hand, we tokenize all words in our training, development and test sets. We get a set of morpheme tokens $\{\boldsymbol{u}_{n,j}\}_{j=1}^{J_n}$ for each word $\boldsymbol{w}_n$, where this word is split into $J_n$ morphological units.

We can now get the optimal length of a morphologically constrained code. With this in mind, we first train a fresh version of our two-stage model on the full set of morphological unit tokens—i.e. $\{\boldsymbol{u}_{n,j} \mid n \le N, j \le J_n\}$, as opposed to the set of full word tokens, $\{\boldsymbol{w}_n\}_{n=1}^{N}$. We estimate the length of this code with the following equation:

$$\begin{aligned}\mathrm{cost}(\mathbb{C}_{morph}) &= \sum_{m\in\mathcal{M}} p(m)\,\left|\mathbb{C}_{morph}(m)\right| \qquad (16)\\ &\lesssim \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{J_n}\left\lceil \log_{|\Sigma|}\frac{1}{p_\theta(\boldsymbol{u}_{n,j})}\right\rceil\end{aligned}$$

Note that this cost estimate is still the average code-length per word token, as such we take the expectation over the meanings distribution. Each word's code-length, though, is now defined as the sum of the length of each of its constituent morphemes.

### 5.3 Graphotactic Constraints

The second linguistic constraint we would like to impose on our codes is graphotactic well-formedness—i.e. we wish our code to be composed only by sequences of characters that comply with the regularities observed in the language, such as e.g. vowel harmony, syllable structure, or word-initial and word-final constraints. We use our generator LSTM for this. As mentioned before, this model is trained on wordforms with dampened frequencies—we thus expect it to learn a language's graphotactic patterns above a minimum quality threshold. We use this character-level model to sample (without replacement) as many unique wordforms as there are word types in that language (see Tab. 3 in App. A).[11] We assign each of these sampled wordforms $\boldsymbol{w}'_n$, ordered by word length, to one of the languages meanings $m_n$, inversely ordered by unigram probability, i.e. $\mathbb{C}_{graph}(m_n) = \boldsymbol{w}'_n$—thus generating an optimally Zipfian frequency–length correlation. With these assignments, we estimate the cost of a graphotactically constrained code:

$$\mathrm{cost}(\mathbb{C}_{graph}) \approx \frac{1}{N}\sum_{n=1}^{N}|\boldsymbol{w}'_n| \tag{17}$$

Analogously, with the generator trained on morpheme units we get an optimal code under both morphological and graphotactic constraints.

$$\mathrm{cost}(\mathbb{C}_{morph+graph}) \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{J_n}|\boldsymbol{u}'_{n,j}| \tag{18}$$

### 5.4 Dataset

We use Wikipedia data in our experiments. The data is preprocessed by first splitting it into sentences and then into tokens using SpaCy's language-specific sentencizer and tokenizer (Honnibal et al., 2020). After this, all punctuation is removed and the words are lower-cased. We subsample (without replacement) one million sentences of each language for our experiments, due to computational constraints. We then use an 80-10-10 split for our training, validation and test sets.

We choose typologically diverse languages for our experiments, each from a different language family: English, Finnish, Hebrew, Indonesian, Tamil, Turkish and Yoruba.[12] These languages vary in their graphotactic tendencies and morphological

[11]Unfortunately, our LSTMs use a softmax non-linearity to assign probabilities and, as such, can't produce zeros. Furthermore, due to the compositional nature of wordform probabilities (see eq. (8)), short implausible forms may have larger probability mass than long plausible ones. To mitigate this effect, when sampling wordforms we impose a minimum threshold of 0.01 on each transition probability $p(\ell_t \mid \boldsymbol{\ell}_{<t})$.

[12]Dataset statistics are presented in App. A.

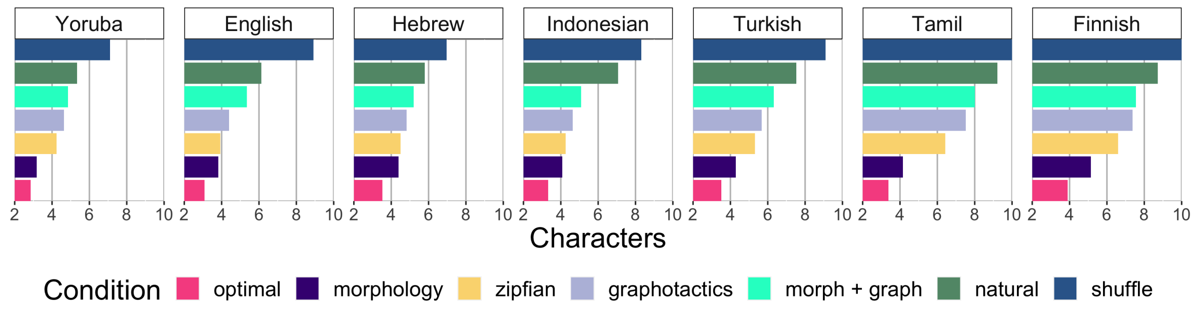

Figure 3: Bar plots of the code lengths under different constraints. In this plot, morphology is constrained through the use of Morfessor segmentation. Length in the shuffle condition for Tamil and Finnish exceed the scale (11.5 and 11 respectively). **Optimal**, **Morph**, **Zipfian**, **Graph**, **Morph + Graph**, **Natural**, **Shuffle**.

complexity. In order to improve our data quality, we hand-defined an alphabet for each language and filter sentences with them, only considering sentences consisting exclusively of valid characters.[13]

### 5.5 Summary

In this paper we consider the following codes:

**Optimal.** An information-theoretically optimal code under our two-stage model, estimated as defined by eq. (15). This is our most compressed code and does not include either morphlogical or graphotactic contraints.

**Morph.** A morphologically constrained code, as defined by eq. (16).

**Graph.** A code constrained by graphotactics, as defined by eq. (17).

**Morph+Graph.** A code constrained by both morphology and graphotactics; defined by eq. (18).

**Natural.** The natural code—equivalent to the average token length and defined by eq. (13). This is the code length actually observed in our corpora.

**Zipfian.** A code estimated by re-pairing wordforms with meanings based on their frequencies; we then compute eq. (13) in this new code. This would be equivalent to the natural code length if lexicons had a perfect word length–frequency correlation (i.e., a Spearman's rank correlation of 1).

**Shuffle.** A code estimated by randomly re-pairing wordforms with meanings and computing eq. (13) in this new code. This would be equivalent to the natural code length if Zipf's law of abbreviation did not exist, i.e. lexicons had no word length–frequency correlation.

[13]We define these sets of valid characters based on Wikipedia entries for the languages and the alphabets available in `https://r12a.github.io/app-charuse/`.

## 6 Results

The average length for each considered code is presented in Fig. 3 and Tab. 1. As expected, we find that the average code length across natural languages is shorter than the shuffle condition and longer than the optimal condition. Interestingly, the codes produced by the other conditions investigated here also have the same identical order across all analyzed languages.

Adding morphological constraints on the code incurs no more than one extra character over the optimal condition—except for Finnish, for which the cost of morphology is slightly above one character. Notably, the use of unsupervised morphological segmentation may introduce some noise into our measurements. Consistently with our expectations, though, Yoruba (a morphologically poor language) pays the smallest cost for its morphology, while Finnish (a morphologically rich one) pays the largest.

BPE and WordPiece systematically produce shorter codes than Morfessor. This is sensible, since the first two would keep most frequent wordforms intact, generating a unique code for each of them. This would lead to codes in which the morphological productivity of frequent and infrequent words differ, amplifying frequency effects encountered in natural languages (Lieberman et al., 2007).

The graphotactic condition yields systematically longer codes than the morphological one, although here there are important differences between languages: English, Hebrew and Indonesian have similar code lengths for both code constraints; in the other languages the graphotactic code is substantially longer than the morphological one.

In all cases, the natural code is longer than the one with both graphotactic and morphological constraints—suggesting languages are not opti-

| Language | Optimal | Morph Morfessor | Morph BPE | Morph WordPieces | Graph | Morph + Graph Morfessor | Morph + Graph BPE | Morph + Graph WordPieces | Zipfian | Natural | Shuffle |
|---|---|---|---|---|---|---|---|---|---|---|---|
| English | 3.09 | 3.82 | 3.34 | 3.31 | 4.39 | 5.34 | 4.67 | 4.70 | 3.93 | 6.11 | 8.91 |
| Finnish | 3.89 | 5.13 | 4.94 | 4.95 | 7.37 | 7.55 | 7.60 | 7.65 | 6.59 | 8.72 | 10.97 |
| Hebrew | 3.52 | 4.38 | 3.98 | 3.99 | 4.82 | 5.19 | 4.95 | 4.88 | 4.50 | 5.79 | 6.97 |
| Indonesian | 3.31 | 4.08 | 3.67 | 3.66 | 4.63 | 5.08 | 5.02 | 4.95 | 4.25 | 7.06 | 8.30 |
| Tamil | 3.38 | 4.15 | 4.07 | 4.01 | 7.52 | 8.01 | 8.16 | 8.22 | 6.41 | 9.21 | 11.48 |
| Turkish | 3.52 | 4.28 | 4.12 | 4.03 | 5.67 | 6.31 | 5.98 | 5.93 | 5.31 | 7.52 | 9.09 |
| Yoruba | 2.84 | 3.18 | 3.00 | 2.97 | 4.63 | 4.85 | 4.69 | 4.61 | 4.24 | 5.34 | 7.10 |

Table 1: The average code lengths under the different coding schemes.

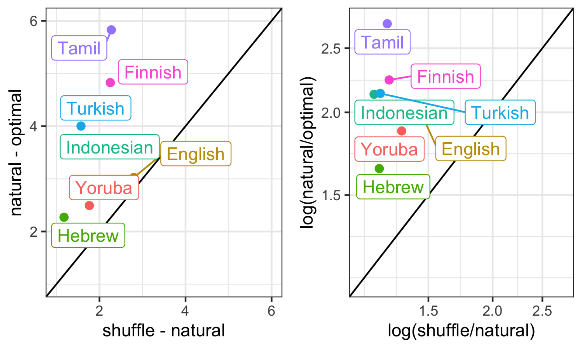

Figure 4: Comparison of the distances—additive (left) and multiplicative (right)—between natural languages and either optimal or shuffled baselines.

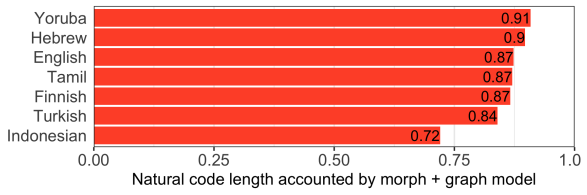

Figure 5: Fraction of code length accounted for by the combined morphology and graphotactics model

mally compressed, even when accounting for these constraints. That said, all of the natural languages are considerably more compressed than a lexicon produced by randomly reassigning wordforms.

## 7 Discussion and Conclusion

In this paper, we introduced a model-based strategy to assess the relative contribution of different constraints on word (code) length at large. In particular, we evaluated how much natural languages differ from systems optimized for Zipf's law of abbreviation. Our proposed model improves upon an old method used to consider the efficiency of the lexicon: **random typing models** (Miller, 1957; Moscoso del Prado, 2013; Ferrer-i-Cancho et al., 2020). Miller introduced the idea of monkeys typing randomly on a keyboard and analyzed the properties of its resulting language. The monkeys' text, however, has no morphological or graphotactic constraints (but see Caplan et al., 2020) and does not follow a language's unigram distribution (Howes, 1968). As such, it cannot directly encode the same meanings or messages as the original language.

Our results show that, while natural languages do tend to map frequent messages to shorter words, the magnitude of this effect varies widely across our set of diverse languages. Notably, the distance between natural languages and the optimal codes is larger than the distance between natural languages and their corresponding shuffled code (see Fig. 4). In other words, natural codes are closer to *not* being optimized (in the Zipfian sense) than to being maximally compressed.

That said, our morphological and graphotactic baselines, when combined, yield codes that display mean code lengths that are (in most cases) closer to the natural code than to the optimal (see Fig. 5). If our models are indeed able to capture the true patterns in our data, then this means that (compositional) morphology and graphotactics, along with the law of abbreviation, are sufficient to account for most of the length of natural codes—as observed in real languages. Graphotactic (primarily) and morphological constraints are enough to derive a code with a similar complexity to that of natural languages, which suggests the other factors discussed above (associated with, e.g., surprisal and non-arbitrary form-meaning mappings) likely play a more modest role in pushing natural languages away from the optimal Zipfian code.

The optimality of the lexicon occupies a major place in the scientific study of the structure and functional evolution of languages (Bentz and Ferrer-i-Cancho, 2016; Gibson et al., 2019; Mahowald et al., 2020). We hope that the method presented here—which allows for a more precise quantification of the (non-)optimality of lexicons—will be used to further the goal of understanding why languages are structured in the ways that they are, while offering insight into the functional trade-offs that underlie language variation and change.

## Ethical Concerns

This paper concerns itself with investigating lexicons' optimality under the perspective of Zipf's Law of Abbreviation. As we focus on computational linguistic experiments, we see no clear ethical concerns here. Nonetheless, we note that Wikipedia (from where we collect data) is not a fully representative source of a language's data—the biases in the data will likely also be present in our results.

## Acknowledgements

Damián E. Blasi acknowledges funding from the Branco Weiss Fellowship, administered by the ETH Zürich. Damián E. Blasi's research was also executed within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'.

## References

András Antos and Ioannis Kontoyiannis. 2001. Convergence properties of functional estimates for discrete distributions. *Random Structures & Algorithms*, 19(3-4):163–193.

Evan Archer, Il Memming Park, and Jonathan W. Pillow. 2014. Bayesian entropy estimation for countable discrete distributions. *Journal of Machine Learning Research*, 15(81):2833–2868.

Matthew Aylett and Alice Turk. 2004. The smooth signal redundancy hypothesis: A functional explanation for relationships between redundancy, prosodic prominence, and duration in spontaneous speech. *Language and Speech*, 47(1):31–56.

Alan Bell, Daniel Jurafsky, Eric Fosler-Lussier, Cynthia Girand, Michelle Gregory, and Daniel Gildea. 2003. Effects of disfluencies, predictability, and utterance position on word form variation in English conversation. *The Journal of the Acoustical Society of America*, 113(2):1001–1024.

Chris Bentz and Ramon Ferrer-i-Cancho. 2016. Zipf's law of abbreviation as a language universal. In *Proceedings of the Leiden Workshop on Capturing Phylogenetic Algorithms for Linguistics*, pages 1–4, Leiden, Netherlands. University of Tübingen.

Damián E. Blasi, Søren Wichmann, Harald Hammarström, Peter F. Stadler, and Morten H. Christiansen. 2016. Sound–meaning association biases evidenced across thousands of languages. *Proceedings of the National Academy of Sciences*, 113(39):10818–10823.

Phil Blunsom, Trevor Cohn, Sharon Goldwater, and Mark Johnson. 2009. A note on the implementation of hierarchical Dirichlet processes. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 337–340, Suntec, Singapore. Association for Computational Linguistics.

Paul Boersma. 1998. *Functional Phonology*. Holland Academic Graphics.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Spencer Caplan, Jordan Kodner, and Charles Yang. 2020. Miller's monkey updated: Communicative efficiency and the statistics of words in natural language. *Cognition*, 205:104466.

Bernardino Casas, Antoni Hernández-Fernández, Neus Català, Ramon Ferrer-i-Cancho, and Jaume Baixeries. 2019. Polysemy and brevity versus frequency in language. *Computer Speech & Language*, 58:19–50.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Isabelle Dautriche, Laia Fibla, Anne-Caroline Fievet, and Anne Christophe. 2018. Learning homophones in context: Easy cases are favored in the lexicon of natural languages. *Cognitive Psychology*, 104:83–105.

Isabelle Dautriche, Kyle Mahowald, Edward Gibson, and Steven T. Piantadosi. 2017. Wordform similarity increases with semantic similarity: An analysis of 100 languages. *Cognitive Science*, 41(8):2149–2169.

Mark Dingemanse, Damián E. Blasi, Gary Lupyan, Morten H. Christiansen, and Padraic Monaghan. 2015. Arbitrariness, iconicity, and systematicity in language. *Trends in Cognitive Sciences*, 19(10):603–615.

August Fenk and Gertraud Fenk. 1980. Konstanz im Kurzzeitgedächtnis - Konstanz im sprachlichen Informationsfluß? *Zeitschrift für Experimentelle und Angewandte Psychologie*, 27(3):400–414.

Gertraud Fenk-Oczlon and August Fenk. 2008. Complexity trade-offs between the subsystems of language. *Language Complexity: Typology, Contact, Change*, 43:65.

Ramon Ferrer-i-Cancho, Christian Bentz, and Caio Seguin. 2020. Optimal coding and the origins of Zipfian laws. *Journal of Quantitative Linguistics*, 0(0):1–30.

Ramon Ferrer-i-Cancho and Michael S. Vitevitch. 2018. The origins of Zipf's meaning-frequency law. *Journal of the Association for Information Science and Technology*, 69(11):1369–1379.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Susanne Gahl. 2008. Time and thyme are not homophones: The effect of lemma frequency on word durations in spontaneous speech. *Language*, 84(3):474–496.

Edward Gibson, Richard Futrell, Steven P. Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. 2019. How efficiency shapes human language. *Trends in Cognitive Sciences*, 23(5):389–407.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2011. Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12:2335–2382.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Davis Howes. 1968. Zipf's law and Miller's random-monkey model. *The American Journal of Psychology*, 81(2):269–272.

David A. Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Dan Jurafsky, Alan Bell, Michelle Gregory, and William Raymond. 2001. Probabilistic relations between words: Evidence from reduction in lexical production. In *Frequency and the Emergence of Linguistic Structure*. John Benjamins, Amsterdam.

Adam King and Andrew Wedel. 2020. Greater early disambiguating information for less-probable words: The lexicon is shaped by incremental processing. *Open Mind*, pages 1–12.

Roger P. Levy and Tim Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. In *Advances in Neural Information Processing Systems*, pages 849–856.

Molly L. Lewis and Michael C. Frank. 2016. The length of words reflects their conceptual complexity. *Cognition*, 153:182–195.

Erez Lieberman, Jean-Baptiste Michel, Joe Jackson, Tina Tang, and Martin A. Nowak. 2007. Quantifying the evolutionary dynamics of language. *Nature*, 449(7163):713–716.

Tamas Linder, Vahid Tarokh, and Kenneth Zeger. 1997. Existence of optimal prefix codes for infinite source alphabets. *IEEE Transactions on Information Theory*, 43(6):2026–2028.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Kyle Mahowald, Isabelle Dautriche, Mika Braginsky, and Edward Gibson. 2020. Efficient communication and the organization of the lexicon. *PsyArXiv preprint PsyArXiv:4an6v*.

Benoît Mandelbrot. 1953. An informational theory of the statistical structure of language. *Communication Theory*, 84:486–502.

George Miller. 1955. Note on the bias of information estimates. *Information Theory in Psychology: Problems and Methods*.

George A. Miller. 1957. Some effects of intermittent silence. *The American Journal of Psychology*, 70(2):311–314.

Padraic Monaghan, Richard C. Shillcock, Morten H. Christiansen, and Simon Kirby. 2014. How arbitrary is language? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1651):20130299.

Fermin Moscoso del Prado. 2013. The missing baselines in arguments for the optimal efficiency of languages. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 35.

Liam Paninski. 2003. Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253.

Steven T. Piantadosi, Harry Tily, and Edward Gibson. 2011. Word lengths are optimized for efficient communication. *Proceedings of the National Academy of Sciences*, 108(9):3526–3529.

Steven T. Piantadosi, Harry Tily, and Edward Gibson. 2012. The communicative function of ambiguity in language. *Cognition*, 122(3):280–291.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2020. Embeddings in natural language processing: Theory and advances in vector representations of meaning. *Synthesis Lectures on Human Language Technologies*, 13(4):1–175.

Tiago Pimentel, Ryan Cotterell, and Brian Roark. 2021a. Disambiguatory signals are stronger in word-initial positions. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

Tiago Pimentel, Rowan Hall Maudslay, Damián Blasi, and Ryan Cotterell. 2020a. Speakers fill lexical semantic gaps with context. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4004–4015, Online. Association for Computational Linguistics.

Tiago Pimentel, Arya D. McCarthy, Damián Blasi, Brian Roark, and Ryan Cotterell. 2019. Meaning to form: Measuring systematicity as information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1751–1764, Florence, Italy. Association for Computational Linguistics.

Tiago Pimentel, Brian Roark, and Ryan Cotterell. 2020b. Phonotactic complexity and its trade-offs. *Transactions of the Association for Computational Linguistics*, 8:1–18.

Tiago Pimentel, Brian Roark, Søren Wichmann, Ryan Cotterell, and Damián Blasi. 2021b. Finding concept-specific biases in form–meaning associations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Virtual. Association for Computational Linguistics.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson–Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

John Sinclair. 1999. A way with common words. *Language and Computers*, 26:157–180.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden. Association for Computational Linguistics.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *COLING/ACL 2006 - 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, volume 1, pages 985–992, Morristown, NJ, USA. Association for Computational Linguistics.

Sean Trott and Benjamin Bergen. 2020. Why do human languages have homophones? *Cognition*, 205:104449.

Rob J. J. H. van Son and Louis C. W. Pols. 2003a. How efficient is speech? In *Proceedings of the Institute of Phonetic Sciences*, volume 25, pages 171–184.

Rob J. J. H. van Son and Louis C. W. Pols. 2003b. Information structure and efficiency in speech production. In *Eighth European Conference on Speech Communication and Technology*, Geneva, Switzerland.

Michael S. Vitevitch and Paul A. Luce. 1999. Probabilistic phonotactics and neighborhood activation in spoken word recognition. *Journal of Memory and Language*, 40(3):374–408.

Greg C. G. Wei and Martin A. Tanner. 1990. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704.

Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. 2020. Consistency of a recurrent language model with respect to incomplete decoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5553–5568, Online. Association for Computational Linguistics.

Yang Xu, Khang Duong, Barbara C. Malt, Serena Jiang, and Mahesh Srinivasan. 2020. Conceptual relations predict colexification across languages. *Cognition*, 201:104280.

George Kingsley Zipf. 1935. *The Psycho-biology of Language: An Introduction to Dynamic Philology*. Houghton Mifflin.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley Press.

## Appendix

## A Dataset sizes

In this section, we present the number of word tokens (Tab. 2) and word types (Tab. 3) in our analyzed datasets.

| | Train | Validation | Test |
|---|---|---|---|
| English | 4,630,371 | 578,510 | 578,796 |
| Finnish | 2,558,634 | 319,546 | 320,716 |
| Hebrew | 4,911,953 | 613,457 | 609,864 |
| Indonesian | 4,039,552 | 506,085 | 507,587 |
| Tamil | 3,286,075 | 412,776 | 412,416 |
| Turkish | 2,676,471 | 333,120 | 332,359 |
| Yoruba | 373,517 | 46,415 | 46,283 |

Table 2: The number of word tokens used in training, validation and testing.

|  | Train | Validation | Test |
|---|---|---|---|
| English | 242,030 | 66,668 | 66,243 |
| Finnish | 466,745 | 109,232 | 110,378 |
| Hebrew | 311,860 | 104,555 | 104,478 |
| Indonesian | 243,118 | 69,792 | 70,079 |
| Tamil | 479,668 | 116,196 | 115,422 |
| Turkish | 308,419 | 84,300 | 83,871 |
| Yoruba | 47,740 | 12,877 | 12,877 |

Table 3: The number of word types used in training, validation and testing.

## B Model Training

As mentioned in the main text, we cannot directly infer the parameters of our model and we use a solution similar to expectation maximization (Wei and Tanner, 1990). We freeze our LSTM generator while learning the PYCRP parameters and then freeze the PYCRP to train the LSTM model.

**Expectation step.** This step uses a Gibbs sampling procedure to estimate the parameters of the PYCRP. For each token in our dataset, we fix all cluster assignments $\mathbf{z}_{-n}$ except for the given token's one $z_n$. We then re-sample this token's cluster based on the marginal probability $p(z_n|\mathbf{z}_{-n}, \boldsymbol{\ell}, \boldsymbol{w}_n)$. We do this for 5 epochs, and use the assignments which result in the best development set cross-entropy. This process can both remove clusters and create new ones by replacing tokens. The set of populated clusters (together with their wordform labels) then allows creating a new wordform dataset of size $K'$, where the distribution of the token frequencies is expected to be less skewed. In practice, this wordform dataset is thus created from the resulting set of cluster labels $\{\boldsymbol{\ell}_k\}_{k=1}^{K'}$, i.e. a word will appear in this new dataset as many times as it was assigned as a cluster label.

**Maximization step.** We use the set of populated cluster labels to train the generator LSTM—assuming that this allows learning a more representative model of a language's graphotactics as the irregular common words are less dominant in its training set. In other words, at each epoch, the generator will be trained in a wordform as many times as it has been assigned as a cluster label.

**Hyperparameters and implementation details.** For the PYCRP, we fix hyper-parameters $a = 0.5$ and $b = 10{,}000$, and we use the optimized Gibbs sampling algorithm designed by Blunsom et al. (2009). As our generator, we use a three layers LSTM with an embedding size of 128, a hidden size of 512 and dropout of .33. This LSTM is trained using AdamW (Loshchilov and Hutter, 2019) and we hotstart it by initially training on the set of word types in the training set (the set of unique wordforms in it).

## C Proof of Proposition 2

We present here the proof of Proposition 2. This proposition is repeated here for convenience:

**Proposition 2.** *If a language model $p(\boldsymbol{w})$ is $\varepsilon$-smooth, then its entropy is finite, i.e.* $\mathrm{H}(W) < \infty$.

*Proof.* To prove this, we will break the entropy of a string in two parts. The entropy of the first character, plus the entropy of the following ones given the first, as in:

$$\mathrm{H}(W) = \mathrm{H}(W_1) + \mathrm{H}(W_{>1} \mid W_1) \tag{19}$$

We first bound the entropy of the first character using a uniform distribution upperbound:

$$\begin{aligned}\mathrm{H}(W_1) &= -\sum_{\boldsymbol{w}_1 \in \Sigma} p(\boldsymbol{w}_1) \log p(\boldsymbol{w}_1) \\ &\leq -\log|\Sigma|\end{aligned} \tag{20}$$

We now use the $\varepsilon$-smoothness property to upperbound the entropy of the following characters:

$$\begin{aligned}&\mathrm{H}(W_{>1} \mid W_1) \\ &\quad= \sum_{\boldsymbol{w}_1 \in \Sigma} p(\boldsymbol{w}_1)\mathrm{H}(W_{>1} \mid W_1 = \boldsymbol{w}_1) \\ &\quad= p(\texttt{EoW})\mathrm{H}(W_{>1} \mid W_1 = \texttt{EoW}) \\ &\qquad+ \sum_{\boldsymbol{w}_1 \in \Sigma, \boldsymbol{w} != \texttt{EoW}} p(\boldsymbol{w}_1)\mathrm{H}(W_{>1} \mid W_1 = \boldsymbol{w}_1) \\ &\quad= \sum_{\boldsymbol{w}_1 \in \Sigma, \boldsymbol{w} != \texttt{EoW}} p(\boldsymbol{w}_1)\mathrm{H}(W_{>1} \mid W_1 = \boldsymbol{w}_1) \\ &\quad\leq \sum_{\boldsymbol{w}_1 \in \Sigma, \boldsymbol{w} != \texttt{EoW}} p(\boldsymbol{w}_1)\mathrm{H}(W_{>1}) \\ &\quad\leq (1-\varepsilon)\,\mathrm{H}(W)\end{aligned} \tag{21}$$

Given both these upperbounds, we can bound the full wordform entropy:

$$\begin{aligned}\mathrm{H}(W) &= \mathrm{H}(W_1) + \mathrm{H}(W_{>1} \mid W_1) \\ &\leq -\log|\Sigma| + (1-\varepsilon)\,\mathrm{H}(W)\end{aligned} \tag{22}$$

Finally, with simple algebraic manipulations we complete the proof:

$$\mathrm{H}(W) \leq -\frac{1}{\varepsilon}\log|\Sigma| \tag{23}$$

□