Searching for the Most Human-like Emergent Language

Brendon Boldt and David Mortensen
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{bboldt,dmortens}@cs.cmu.edu

Abstract

In this paper, we design a signalling game-based emergent communication environment to generate state-of-the-art emergent languages in terms of similarity to human language. This is done with hyperparameter optimization, using XferBench as the objective function. XferBench quantifies the statistical similarity of emergent language to human language by measuring its suitability for deep transfer learning to human language. Additionally, we demonstrate the predictive power of entropy on the transfer learning performance of emergent language as well as corroborate previous results on the entropy-minimization properties of emergent communication systems. Finally, we report generalizations regarding what hyperparameters produce more realistic emergent languages, that is, ones which transfer better to human language.

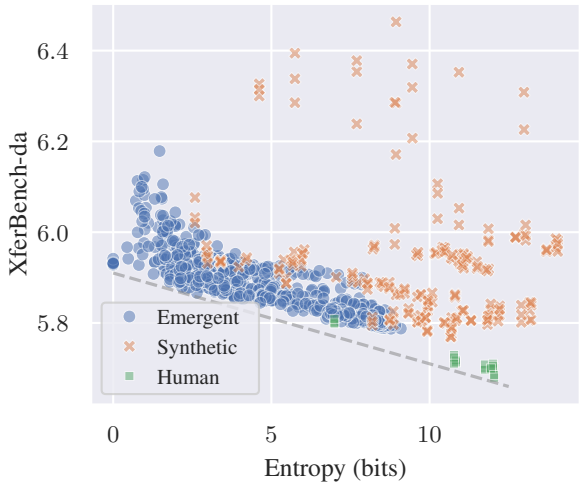

Figure 1: Hyperparameter search shows that emergent and human languages tend towards the Pareto frontier of minimizing entropy and minimizing XferBench score (lower is better) while non-emergent synthetic languages less reliably follow this trend. Dashed gray line represents a lower bound on entropy versus XferBench score.

1 Introduction

Emergent language has tremendous potential to generate realistic human language data for deep learning methods without the need to collect data directly (or indirectly) from humans (Boldt and Mortensen, 2024c). This stems from the fact that emergent language aims to replicate the communicative pressures that drive the development of human language and are hypothesized to explain various patterns observed in linguistics (Scholz et al., 2024). Yet little work has been done to date designing emergent communication systems to generate languages with high statistical similarity to human languages. Such languages could better serve as synthetic human language data for pretraining and evaluating NLP models. Thus, in this paper, we generate emergent languages with a signalling game that have a high degree of similarity to human languages, demonstrating state-of-the-art performance on emergent-to-human language deep transfer learning. Specifically, we use Bayesian hyperparameter search to optimize a signalling game on the XferBench benchmark (Boldt and Mortensen, 2024b).

Producing emergent languages which are more realistic (i.e., similar to human language) is one of the core goals of the field as a whole since the utility of emergent language is often predicated on its resemblance to human language (Boldt and Mortensen, 2024c). This paper takes a direct, principled approach to this goal by finding hyperparameters which maximize an emergent languages similarity to human language from a statistical perspective. Such an approach is in stark contrast to a more arbitrary approach to selecting hyperparameters which is common in the methods of emergent communication. For example, vocabulary sizes in emergent languages are often very small (only one of eight emergent language environments surveyed in Boldt and Mortensen (2024a) exceeds

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 23289–23307
November 4-9, 2025

a vocabulary size of 70) while our research suggests that the optimal vocabulary size is in the 1k to 10k range. Increasing vocabulary sizes, then, not only improves transfer learning performance but also makes it possible for emergent languages to replicate the long-tailed, Zipfian word distribution that is characteristic of human language (Zipf, 1949; Piantadosi, 2014), for example. We produce a handful of such hyperparameter recommendations based on our empirical evaluations.

Beyond these recommendations, our experiments also confirm a significant relationship between transfer learning performance and corpus entropy. Not only does it appear that the entropy of a corpus determines a lower bound on XferBench score (lower is better) but that emergent languages minimize entropy with respect to a given XferBench score in a way that procedurally generated (i.e., non-emergent, synthetic) languages do not (see Fig. 1). Such minimization is, significantly, an *emergent* phenomenon as neither entropy nor transfer learning performance are directly involved in the optimization of the emergent communication system (and neither entropy nor XferBench incorporate each other). This observation is significant in two regards: First, it suggests that transfer learning and, consequently, statistical similarity to human language can be (partially) explained with information theory. Second, it aligns closely with prior work that finds that emergent communication minimizes entropy with respect to task success within the environment (Kharitonov et al., 2020; Chaabouni et al., 2022).

We discuss related work in Section 2. Methods are discussed in Section 3, and the experiments are presented in Section 4. An analysis of the results is performed in Section 5 with discussion and conclusion in Sections 6 and 7.

Contributions We (1) introduce emergent communication environments which produce the most human language-like emergent languages to date, as shown by state-of-the-art performance on a deep transfer learning task using the XferBench benchmark; (2) provide concrete recommendations on better hyperparameter settings for emergent communication experiments so as to make them more statistically similar to human language; and (3) provide evidence that entropy minimization is a general property of emergent communication systems, finding that it is minimized with respect to transfer learning performance.

2 Related Work

At a high level, emergent communication (also called *emergent language*) combines natural language processing, deep multi-agent reinforcement learning, and linguistics to study how natural language-like communication systems evolve or emerge from scratch. One of the primary aims of this field is to discover what features of human language (e.g., compositionality) emerge from the environment and learning dynamics of the agents. For a general overview of deep learning-based emergent communication research, see Lazaridou and Baroni (2020). For the most part, this paper does not have any directly related work as optimizing emergent languages themselves across multiple game instances is relatively unexplored. Below we present some particular facets of this paper which overlap with prior work.

This paper shares the goal of producing emergent language corpora that are suitable for transfer learning to human languages with Yao et al. (2022), although Yao et al. (2022) do not optimize the emergent languages directly and focus on validating the *corpus transfer* technique (i.e., the basis of XferBench). Boldt and Mortensen (2023), similarly to this paper, investigate the effect of hyperparameters on emergent communication, although their study focuses primarily on mathematically analyzing and explaining the effects rather than optimizing the emergent language for an evaluation metric. Finally, this paper scales up emergent communication game hyperparameters in a way that overlaps with Chaabouni et al. (2022), although the latter focuses on addressing the practical challenges of scaling up certain facets of the signalling game (e.g., number of agents) rather than directly optimizing for a particular objective.

The task of generating emergent languages for pretraining NLP models falls within the broad category data augmentation with synthetic data but differs from most other approaches due emergent language's unique nature as an *emergent* phenomenon. First, emergent language differs from procedurally generating data from rules because emergent techniques preclude stipulating the exact process for generating the data; expert knowledge is incorporated into designing the system which generates the data, not generating the data itself. On the other hand, emergent language differs from using pretrained language models to generate synthetic data since emergent communication is derived from

scratch, again precluding any (pre)training on human language data.

3 Methods

3.1 Objective: XferBench

The ultimate objective that we are optimizing for is transfer learning performance on downstream human language tasks. This objective is quantified by XferBench (Boldt and Mortensen, 2024b, MIT license), which measures how much pretraining on an emergent language corpus decreases cross-entropy on a limited-data, downstream language modelling task on human languages (illustrated in the gray box of Fig. 2). While language modeling performance does not capture every aspect of mastery of language, it does serve as the backbone of many NLP tasks (e.g., generative models, automatic speech recognition, machine translation). From a practical point of view, language modeling is also one of the simpler and less expensive downstream tasks to test on (cf. testing on machine translation in Boldt and Mortensen (2024b)).

Since the output of XferBench is mean cross-entropy across human languages, a lower score better. XferBench takes as input a corpus of 15 million tokens, which is used for the pretraining stage and finetunes on 2 million tokens for each evaluation (human) language. The language model used for XferBench is based on GPT-2 (Radford et al., 2019) and has ∼60 million parameters. Since XferBench has a long runtime, we use a modified version only during hyperparameter search termed *XferBench-da* which only evaluates on one human language (viz. Danish) which we found to have high correlation ($R^2 > 0.95$) with the complete XferBench; see Appendix A for details.

3.2 Environment: signalling game

The environment we use in our experiments is the signalling game. In particular we use the discrimination variant of the signalling game based on the implementation in EGG (Kharitonov et al., 2021, https://github.com/facebookresearch/EGG, MIT license). The discrimination variant of the signalling game consists of two agents, a sender and a receiver interacting for a single round. In a given round, the sender observes an input, sends a message to the receiver, and the receiver selects an observation out of a number of candidates based on the message. Of the candidate observations, one is correct (i.e., the same as the sender's input), and the rest are "distractors". In the implementation used in this paper:

- Observations are concatenations of a fixed number of discrete-valued vectors (see Appendix B for details).
- Messages are sequences of integers represented by one-hot vectors.
- Agents are feed-forward neural networks with one hidden layer and GRU-based RNNs to generate/read the message.[1]
- The sender–receiver system is trained end-to-end with backpropagation using a Gumbel-Softmax layer (Maddison et al., 2017; Jang et al., 2017) to generate the message.

Overall, this emergent communication system is about as "vanilla" as is studied in the literature. This is advantageous for a number of reasons:

- The environment is fast to run, requiring 10 to 120 minutes depending on the hyperparameters.
- It has a (comparatively) limited number of hyperparameters making hyperparameter search more tractable and reducing potential confounding variables.
- It serves as a "lower bound" for optimizing emergent communication environments since we can determine the maximum performance possible in a system with minimal complexity.
- The training is stable, converging to a high success rate for most hyperparameter combinations.

The data is generated for the input corpus to XferBench by sampling from the dataset of observations and feeding these observations into the sender which generates the message.

3.3 Variables: hyperparameters

The hyperparameters are the independent variable of the primary experiments presented in this paper; that is, the hyperparameters will be varied in order to optimize the system for the objective function. Some hyperparameters manipulated in this study are unique to the signalling game (e.g., how many attributes and values in the signalling game observations) while others come from deep learning-based architectures more generally (e.g., learning rate, neural network architecture).

We primarily investigate the following hyperparameters:

[1] Other architectures were investigated in a follow-up experiment described in Appendix K.

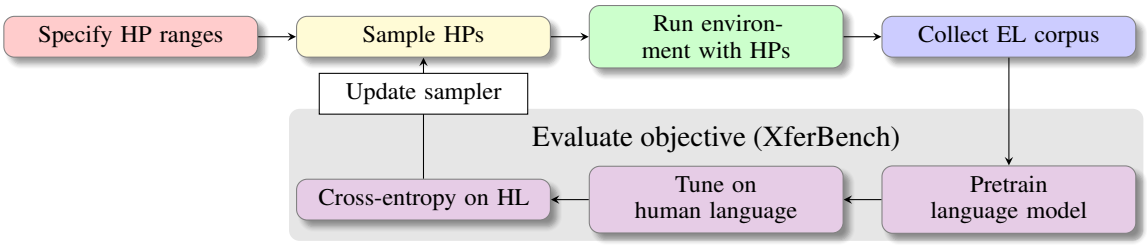

Figure 2: Illustration of hyperparameter optimization with XferBench (adapted from Boldt and Mortensen (2024b) (CC BY 4.0 License)).

Learning rate Multiplication factor for the weight updates for parameters in the neural network.

Embedding size Size of embedding layer in both the sender and the receiver networks; these are independent layers, but their sizes are varied in unison for hyperparameter search.

Hidden size The size of hidden layer in both the sender and the receiver networks; values are varied in unison.

***n* attributes** Number of one-hot vectors in each observation.

***n* values** Size of one-hot vectors in observations.

***n* distractors** Number of incorrect observations shown to the receiver (in addition to the correct one).

***n* epochs** Number of training examples seen.[2]

Temperature Temperature of the Gumbel-Softmax layer which the sender uses to generate messages during training.

Vocabulary size Dimension of the one hot vectors which comprise the message.

Message length Number of one-hot vectors in a message.[3]

Other hyperparameters that were either not discussed or not investigated are documented in Appendix C. Although this set of hyperparameters only covers a small portion of the possible variations of the signalling game (let alone other emergent language games), it covers many basic hyperparameters which show up commonly in emergent communication research.

3.4 Optimization: hyperparameter search

Finally, we discuss the method used for optimizing the hyperparameters of the emergent communication system (the parameters system itself are optimized with backpropagation, as mentioned above). The simplest of all hyperparameter search methods is grid search, where each element of the Cartesian product of every set of hyperparameter values is evaluated. Even using a modest 3 values per aforementioned hyperparameter would require $3^{10} \approx 60\,000$ trials, taking 5 GPU-years (at 1 hour per trial). Thus, we employ Bayesian parameter optimization to more efficiently select hyperparameter combinations to evaluate; this additionally allows us to specify a range of hyperparameter values instead of individual values. This process is illustrated in Fig. 2.

We specifically use a Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) as implemented in Optuna (Akiba et al., 2019, MIT license). At a basic level, TPE works by partitioning hyperparameter combinations into a "good" set and a "bad" set based on the objective function value and selects the next combination of hyperparameters by maximizing the probability of the hyperparameters being in the good set divided by the probability of them being in the bad set. These probability estimates use multivariate kernel density estimators and permit discrete, categorical, and conditional hyperparameter values. After running the environment with the hyperparameters and the objective function on the result, the sampler's probability estimates are updated in accordance with the objective function's value. For a more detailed explanation, see Watanabe (2023).

4 Experiments

The code to run the experiments and analyses is publicly available at `https://github.com/brendon-boldt/signalling-game-search` under the MIT license.

4.1 Hyperparameter searches

In this paper, we present four main searches (Searches 1–4) with two additional searches

[2] Since the data is procedurally generated, a new dataset of 1024 observations is sampled for each epoch.

[3] Technically, the implementation allows for variable length messages, but optimization led to all messages always being the max length.

#	|Trials|	|Attrs.|	|Vals.|	|Distrs.|	Temp.	|Embed.|	|Hidden|	LR	|Vocab|	Length	|Epochs|
1	578	[3, 7]	[3, 7]	[1, 127]	[0.1, 10]	[8, 128]	[8, 128]	[500µ, 50m]	[10, 20k]	[1, 40]	500
2	171	[5, 10]	[5, 10]	—	[0.5, 4]	[64, 512]	[64, 512]	[500µ, 5m]	[300, 30k]	—	—
3	140	—	—	—	—	—	—	—	—	—	[500, 5k]
4	282	[6, 20]	6	23	2	128	256	[1m, 3m]	[500, 30k]	—	—
4*	1	11	6	23	2	128	256	1.79m	9721	16	1715

Table 1: All hyperparameters were treated as log-scale hyperparameters. $|\cdot|$ refers to cardinality. "—" means unchanged from the previous run. µ, m, and k refer to the SI prefixes micro ($\times 10^{-6}$), milli ($\times 10^{-3}$), and kilo ($\times 10^{3}$), respectively.

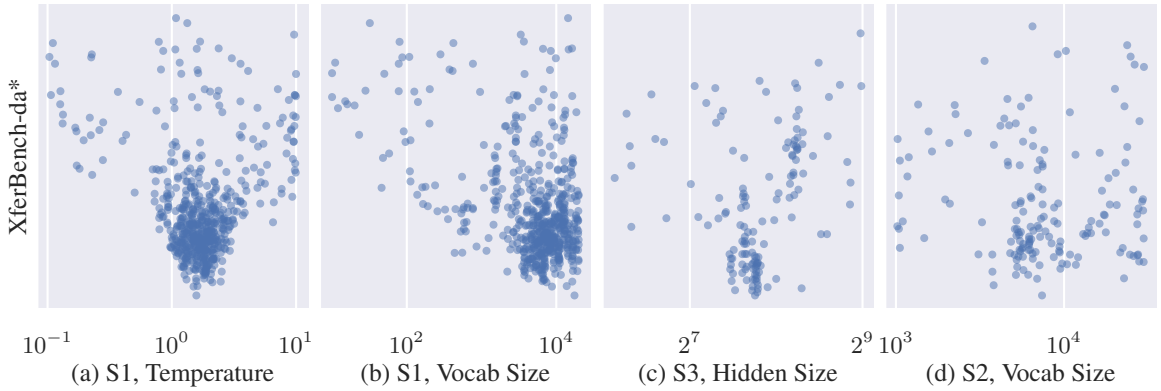

Figure 3: Examples of different hyperparameter–objective relations observed in the various searches and hyperparameters. From left-to-right, we have: (a) a clear best value, (b) a clear trend outside the provided range, (c) a weak trend toward a particular value, and (d) no definite trend. The y-axis based on different "sizes" of XferBench-da normalized to similar scales.

(Searches 5r and 6e) for use in later analyses (Section 5). The following is a summary of the hyperparameter searches:

Search 1 Large number of hyperparameters varied with a wide range; used small version of XferBench-da (1M train tokens for 1 epoch, 200k test tokens for 2 epochs).

Search 2 Same number of hyperparameters varied with smaller or larger ranges depending on results of Search 1; used medium version of XferBench-da (4M train tokens for 2 epochs, 1M test tokens for 3 epochs)

Search 3 Same parameters as Search 2 while allowing number of epochs to go higher and using the full version of XferBench-da (15M train tokens for 5 epochs, 2M test tokens for 10 epochs).

Search 4 Reduces ranges or fixes parameters from Search 3 to maximize exploitation of good parameters; 4* in Table 1 is the best-performing trial from Search 4.

Search 5r Most parameters varied with wide ranges except using *random sampling* to remove sampling bias; similar to Search 1 with narrower ranges on learning rate. Discussed in Section 5.2.

Search 6e Optimized for maximizing entropy after a number of previous searches (not discussed in the paper); similar to Search 4 in this regard. Discussed in Section 5.2.

The parameters of Searches 1–4 are given in Table 1 (for complete table, see Table 3). The implementation defaults for other hyperparameters were used unless otherwise specified. Optuna's default parameters for TPE were used across all experiments.

The signalling game takes 5 to 40 minutes to run (depending primarily on the number of epochs, and, to a lesser extent, the message length), and the full version of XferBench-da takes approximately 40 minutes to run. Thus, the average trial (for the latter searches) takes approximately [0.75, 1.5] hours. Parallelization was used to run multiple trials within a search at a time. See Appendix E for a discussion of computing resources used.

Search design For each iteration of the primary searches (i.e., 1–4), we changed the search parameters based on their correlation with the objective function. We observed four main univariate pat-

terns[4], illustrated in Fig. 3. For parameters with a clear trend toward the center (Fig. 3a), we narrowed the range to encourage exploiting good values. Some parameters trended to one side of the range (Fig. 3b), which indicated needing to extend the range. Parameters with weak to no trend (Figures 3c and 3d) were left unchanged for the initial searches and given an arbitrary value for the final search to reduce noise. Full hyperparameter plots given in Appendix J.

Searches 1 and 2 used a reduced version of XferBench to execute more trials quickly and prune the less promising hyperparameter ranges; nevertheless, caution was exercised in pruning since scaling up XferBench could change optimal hyperparameter values. The irregular number of trials per search were due to executing as many trials as possible within a certain time (rather than aiming for a particular number of trials).

4.2 Languages evaluated

We select three categories of languages to evaluate with XferBench: human languages, those generated with the hyperparameter search discussed above, and extant emergent language corpora from ELCC (Boldt and Mortensen, 2024a, `https://huggingface.co/datasets/bboldt/elcc`, CC BY 4.0). The primary goal is for the search-derived languages to outperform all existing emergent languages and get as close to human language performance as possible. For the human languages, we use a subset of the baselines provided in Boldt and Mortensen (2024b). In particular, we use Mandarin and Hindi because they were the best- and worst-performing human languages, respectively, and French and Arabic to round out the language families represented.

For the search-derived languages, we selected the three best languages from the final primary run of hyperparameter search (Search 4) and evaluate them on the full set of evaluation languages in XferBench. We additionally include the three highest-entropy languages from the entropy-maximizing search (Search 6e, discussed further in Section 5.2).

Finally, for the emergent language-based points of comparison, we select three of the best performing languages from ELCC. Most notably, this includes Yao+ (`corpus-transfer-yao-et-al/coco_2014` (Yao et al., 2022)) which performed far better than all other emergent languages on XferBench. Mu+ (`generalizations-mu-goodman/cub-reference` (Mu and Goodman, 2021)) and Chaabouni+ (`ec-at-scale/imagenet-10x10` (Chaabouni et al., 2022)) were also included as more typical high-performing emergent languages on XferBench.

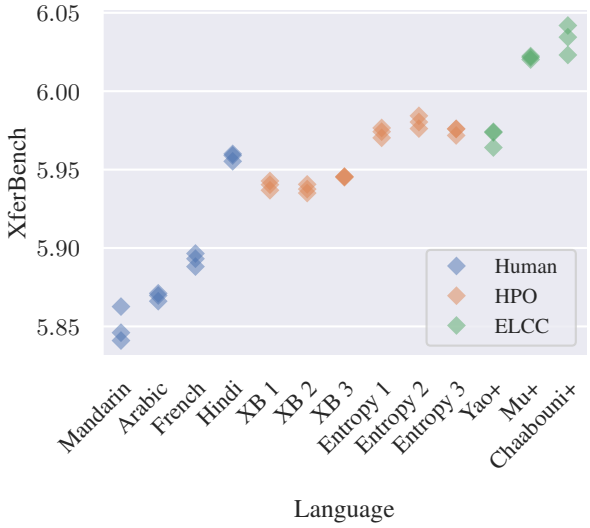

Figure 4: Plot of XferBench scores on emergent and human languages. XB 1–3 are emergent language corpora derived from Search 4 and Entropy 1–3 from Search 6e.

4.3 Results

Figure 4 shows 3 randomly seeded runs of the full XferBench score for each corpus. For the emergent languages from hyperparameter search, the models were restored from checkpoints saved during the search, but the corpora were generated independently of the search. First, we see that the emergent languages from the XferBench-based search (XB 1–3) outperform all other emergent languages and even the Hindi corpus[5]. While it is indeed significant that these emergent languages outperform a human language corpus, this corpus is also an outlier, and the emergent languages are still relatively far from matching the performance of the rest of the human language corpora. Nevertheless, these figures show that the XB 1–3 languages achieve state-of-the-art levels of similarity to human language. The corpora from the entropy-based search (Entropy 1–3) perform well, comparably to Yao+, but significantly worse than the XferBench-search languages.

[4]While we did look for multivariate effects (i.e., hyperparameters that are *not* independent), we did not observe any notable trends.

[5]For a brief discussion of Hindi's poor performance, see Appendix F.

5 Analysis

5.1 Importance of hyperparameters

Vocabulary size The most notable hyperparameter trend we found was with vocabulary size, where the best-performing languages had unique token counts of on the order of 1000 and vocabulary sizes closer to 10 000 (see Fig. 11); that is, the model could use up to 10 000 unique words but only uses 1000 after training. For reference, it is common practice in emergent communication research to use vocabulary sizes well under 100 (e.g., only 1 out of the 8 systems in ELCC produce corpora with >70 unique tokens).

Scaling up Similarly to vocabulary size, we observe indications to scale up message length, neural network layer size, and task information (i.e., number of attributes, values, and distractors): the most human like emergent languages require longer training, larger networks, and higher-information tasks compared to common practice in the emergent communication literature. Along with vocabulary size, these hyperparameter are most often trivial to adjust, meaning there is little reason not to adjust standard practice in emergent communication research to using hyperparameters in these ranges.

Learning rate Finally, in terms of raw importance with respect to XferBench score, learning rate was most significant; this result is not surprising as learning rate is significant in any deep learning algorithm. Nevertheless, part of the difficulty with learning rate is that there is no one best learning rate, and so performing at least some hyperparameter tuning with learning rate will be necessary for optimal performance.

Summary of recommendations We recommend the following hyperparameters as a rule of thumb: vocabulary size: 10 000, hidden layer size: 256, embedding layer size: 128, message length: 20, observation diversity: the higher the better (e.g., $6^{12} \approx 2$ trillion unique observations), epochs: train until task success plateau (not just until arbitrary threshold), learning rate: tune on final setting, neural architecture: 2-layer LSTM with 2 hidden layers[6].

[6]Based on follow-up experiments in Appendix K.

5.2 Entropy and XferBench

The most striking correlation we observe in our experiments is between XferBench score and unigram token entropy, which is illustrated in Fig. 1 (Pearson's $r = -0.57$ for Search 5r only). The emergent languages pictured are all those generated by Searches 4 and 5r, while the human languages are taken from Boldt and Mortensen (2024b). We see that low entropy languages tend to score poorly on XferBench while high scoring languages have higher entropy; this aligns with the observed correlation between XferBench and entropy in Boldt and Mortensen (2024a). Furthermore, this correlation follows the same trend we see in human languages with respect to entropy.

Entropy's lower bound In particular, we have illustrated a lower bound of low entropy–low XferBench score that describes both emergent and human languages (the gray dashed line in Fig. 1). This suggests that given a certain entropy, there is a hard limit on the performance XferBench that can be achieved. While further theoretical and empirical analysis would be required to verify that this a true lower bound, this aligns with the notion of language models as entropy-minimizers: Language models, in order to reduce the entropy on a target language, require a certain degree of entropy (i.e., information) in the pretraining data. Hence, low-entropy, low-information pretraining data leads to language models which reduce entropy less (i.e., yielding higher cross-entropy).

Entropy minimization Looking again at Fig. 1, we also see that the high-entropy, high-XferBench quadrant (upper right) is also sparsely inhabited. In fact, emergent and human languages seem to lie primarily near the Pareto frontier of low-entropy, low-XferBench score mentioned above. This comes in contrast to the XferBench scores of a variety of synthetic languages (descriptions of which are given in Appendix G) which often do not demonstrate this Pareto efficiency, even for synthetic languages performing well on XferBench.

This result is concordant with the related claim that entropy is "minimized" inside of emergent communication systems (Kharitonov et al., 2020; Chaabouni et al., 2021). Such work has shown that emergent communication systems tend to find Pareto efficient solutions in terms of maximizing task success and minimizing entropy (this correlation in the hyperparameter search is discussed

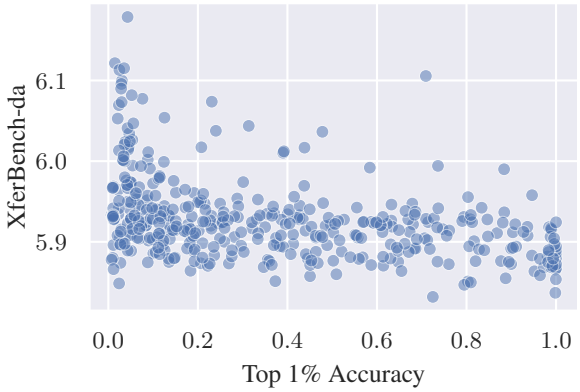

Figure 5: Accuracy versus XferBench for Search 5r. Accuracy is measured as proportion of rounds for which the correct observation is ranked in the top-1 percentile among all distractors.

briefly in Appendix H).

Optimizing on entropy directly The correlation between entropy and XferBench naturally leads to a potential performance improvement: Why not use entropy as the hyperparameter objective instead of XferBench? Entropy takes seconds to compute instead of close to an hour. This is the experiment performed in Search 6e which was successful in producing languages with good XferBench scores but which still performed significantly worse than optimizing on XferBench directly (see Fig. 4).

Given that the lower bound of entropy versus XferBench score is tighter than the upper bound, it is roughly the case that low entropy implies poor XferBench performance, but high entropy does not necessarily imply good XferBench performance. Furthermore, it is also possible that optimizing directly for entropy results in degenerate solutions that find trivial or otherwise unhelpful ways to boost entropy. Thus, the fact that the entropy-based search finds good but not optimal emergent languages fits with the earlier observation about bounds of entropy and XferBench score. With these observations in mind, a refinement to the hyperparameter search algorithm would be to prune low-entropy trials before running XferBench while fully evaluating the trial on XferBench if it has high entropy.

Task success The correlation between task success and XferBench score (Fig. 5, Pearson's $r = -0.40$) is not as dramatic as with entropy. Nevertheless, the negative correlation (better task success, better XferBench score) matches the expectation that the realism of emergent language is positively correlated with the efficacy of the language. This relationship is a foundational assumption of emergent communication techniques generally: the realism of simulation-derived language comes, in part, from its development out of the functional pressures to communicate. Thus, if the emergent communication does not function well, we would not have reason to think it would be similar to human language, absent evidence.

6 Discussion

Similarity to human language The primary motivation for optimizing emergent communication systems on XferBench is to create more human language-like emergent languages. In this way, this environment and the recommended hyperparameters provide a better baseline environment for future emergent communication research to work from. This similarity to human language is critical for nearly every application of emergent communication research, not only related to machine learning and NLP but also areas with more linguistic focus (Boldt and Mortensen, 2024c). Although XferBench quantifies a decidedly more deep learning, data-driven notion of similarity, this account is complimentary with more explicitly linguistic notions of similarity to human language.

For example, linguistic phenomena such as parts of speech fundamentally concern whole classes of words behaving predictably in a variety of environments. Thus, trivially small languages are not suitable for addressing such phenomena as there are not classes of words and no variety to generalize over. Even something as fundamental as the Zipfian distribution of words in human language presupposes a large vocabulary size (Zipf, 1949; Piantadosi, 2014).[7] Furthermore, smaller-scale emergent languages are a greater risk for overfitting since the capacity of a neural network quickly enters the overparameterization regime when the language has as small vocabulary, message length, etc. (Gupta et al., 2020).

Emergent properties The relationship between entropy, task success, and XferBench score demonstrated in the hyperparameter searches emphasizes the presence of *truly emergent* properties and processes in emergent communication: Neither entropy nor transfer learning performance are directly

[7] A follow-up experiment in Appendix I shows that even high-performing emergent languages from our experiments have a decidedly non-Zipfian distribution.

optimized for (cf. task success). Just as Pareto efficient entropy has been found for task success in emergent languages (Kharitonov et al., 2020), we find some degree of Pareto efficiency with entropy and XferBench performance (and to a limited degree with task success and XferBench). What this shows is that the communicative pressures and information theoretic considerations are a key ingredient in emergent language's similarity to human language. Thus, task success and entropy serve as additional ways to reason about emergent language and how to apply it to human language. Nevertheless, the limited correlation we find among these properties also tells us that emergent language is not trivially explained by these factors either.

Future work On the front of creating more human language-like emergent languages, a next step is to introduce new variations of the signalling game, entirely new environments, or more sophisticated neural architectures and optimize them on a metric like XferBench in order to progress towards the long-term goal of producing realistic emergent languages for transfer learning. Because this paper has wrung as much performance as is possible from the basic signalling game environment, there can be greater certainty that innovations producing higher-performing languages are actually causing the improvement. Otherwise, more trivial factors like better learning rate tuning could become confounding variables.

As far as investigating the entropy minimization pressure in emergent languages, further theoretical work needs to build models and generate testable hypotheses; theoretical models are the key to scientific explanation beyond merely showing the existence of correlations. Nevertheless, this paper has shown that hyperparameter turning can be an effective tool for producing a large variety of emergent language that preclude hyperparameters being confounding variables. Such methods of generating datasets will be invaluable in empirically testing theoretical models of emergent language.

7 Conclusion

In this paper we have used hyperparameter search to generate the most human language-like emergent language to date, as quantified by XferBench. Not only does this represent a step forward for using emergent languages as realistic synthetic data for transfer learning but also provides insight into how hyperparameters can be better addressed in future emergent communication research. Finally, the hyperparameter search reveals further importance of the role of entropy in emergent language. High entropy appears to be a necessary condition for good transfer learning performance while at the same time, emergent language appears to minimize entropy for a given level of transfer learning performance. Furthermore, this entropy minimization is not replicated in synthetic languages suggesting that emergent language is more than just "synthetic languages with extra steps".

Limitations

In terms of finding the most human language-like emergent language, this study is limited in terms of the simplicity of the environment and agent design. A single round signalling game with a fixed sender and receiver and uniform, synthetic observations is a no-frills environment which, while good for stability and simplicity, is limited in the richness of information to be communicated, and as a result, the languages it can produce. Thus, while the presented insights can apply, in part, to many settings, it does not come close to providing a comprehensive account of the effects of hyperparameters in emergent communication.

Regarding the investigation of the link between entropy and XferBench score and task success, we were not able to build any theoretical models to scientifically test particular hypotheses about the relationships between the variables; instead, we are only able to offer empirical evidence that there are trends warranting further investigation. Finally, the recommendations we can given regarding the hyperparameters of emergent communication systems are limited because hyperparameter search is relatively "messy"; it is geared toward maximizing performance more than uncovering generalizable trends. Additionally, we perform our experiments with a signalling game which provides only limited evidence for the behavior of emergent communication systems with different tasks.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and

Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Brendon Boldt and David Mortensen. 2023. Mathematically modeling the lexicon entropy of emergent language. *arXiv*, 2211.15783.

Brendon Boldt and David Mortensen. 2024a. ELCC: the Emergent Language Corpus Collection. *Preprint*, arXiv:2407.04158.

Brendon Boldt and David Mortensen. 2024b. XferBench: a data-driven benchmark for emergent language. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1475–1489, Mexico City, Mexico. Association for Computational Linguistics.

Brendon Boldt and David R Mortensen. 2024c. A review of the applications of deep learning-based emergent communication. *Transactions on Machine Learning Research*.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. 2021. Communicating artificial neural networks develop efficient color-naming systems. *Proceedings of the National Academy of Sciences*, 118(12):e2016569118.

Rahma Chaabouni, Florian Strub, Florent Altché, Eugene Tarassov, Corentin Tallec, Elnaz Davoodi, Kory Wallace Mathewson, Olivier Tieleman, Angeliki Lazaridou, and Bilal Piot. 2022. Emergent communication at scale. In *International Conference on Learning Representations*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Abhinav Gupta, Cinjon Resnick, Jakob Foerster, Andrew Dai, and Kyunghyun Cho. 2020. Compositionality and capacity in emergent languages. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 34–38, Online. Association for Computational Linguistics.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. 2020. Entropy minimization in emergent languages. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5220–5230. PMLR.

Eugene Kharitonov, Roberto Dessì, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. 2021. EGG: a toolkit for research on Emergence of lanGuage in Games. https://github.com/facebookresearch/EGG.

Angeliki Lazaridou and Marco Baroni. 2020. Emergent multi-agent communication in the deep learning era. *Preprint*, arXiv:2006.02419.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Jesse Mu and Noah Goodman. 2021. Emergent communication of generalizations. In *Advances in Neural Information Processing Systems*.

S.T. Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychon Bull Rev*, 21:1112—1130.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Barbara C. Scholz, Francis Jeffry Pelletier, Geoffrey K. Pullum, and Ryan Nefdt. 2024. Philosophy of Linguistics. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Spring 2024 edition. Metaphysics Research Lab, Stanford University.

M.P. Schützenberger. 1963. On context-free languages and push-down automata. *Information and Control*, 6(3):246–264.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

Shuhei Watanabe. 2023. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv*, 2304.11127.

Shunyu Yao, Mo Yu, Yang Zhang, Karthik R Narasimhan, Joshua B. Tenenbaum, and Chuang Gan. 2022. Linking emergent and natural languages via corpus transfer. In *International Conference on Learning Representations*.

GK Zipf. 1949. *Human behavior and the principle of least effort*. Addison-Wesley, Cambridge, MA.

	All	Human	Emergent
Basque	0.340	0.685	0.318
Danish	0.992	0.966	0.987
Finnish	0.971	0.968	0.969
Hebrew	0.967	0.967	0.977
Indonesian	0.988	0.952	0.983
Japanese	0.973	0.930	0.974
Kazakh	0.983	0.936	0.977
Persian	0.972	0.951	0.971
Romanian	0.985	0.945	0.982
Urdu	0.951	0.849	0.929

Table 2: R^2 values for individual target XferBench languages predicting the full XferBench score. *Human* and *Emergent* refer to the R^2 value considering only the human or emergent languages, respectively.

A Correlation of Evaluation Languages

One of XferBench's chief weaknesses is its long runtime, taking 2 to 6 hours depending on the GPU used. Approximately 30% of that time is spent on the initial pretraining with the emergent language corpus, with the other 70% spent on finetuning and testing on the 10 downstream languages. We observe from the XferBench scores on the emergent languages of ELCC and the human language baselines of Boldt and Mortensen (2024b) that 9 out of the 10 evaluation languages are highly correlated with each other, that is, the XferBench score on one language is highly predictive of the overall XferBench score. In particular, test cross-entropy on Danish (da) alone can predict >95% of the variation of the overall XferBench score (i.e., the linear regression has an $R^2 > 0.95$). For this reason, in the hyperparameter optimization trials, we compute XferBench-da (XferBench evaluated on Danish only) which is around 3× faster than the full XferBench; the final evaluation nevertheless uses the full set of evaluation language for XferBench.

In Table 2, we show the R^2 values derived from training a linear model on just one of the target language's XferBench scores to predict the overall XferBench score. The emergent languages are all of the corpora from ELCC (Boldt and Mortensen, 2024a), and the human language corpora are the baselines from the original XferBench paper (Boldt and Mortensen, 2024b). R^2 value corresponds to the percent of the variance in the full XferBench score explained by just the score (i.e., cross-entropy) on that particular target language. We find, strikingly enough, that all of the target languages, with the exception of Basque, are highly correlated, having R^2 values above 0.95 all languages, and greater than 0.80 even when considering human languages alone. Danish, of all of the languages, has the highest R^2 value (>0.99), which is the reason we select it as the sole target for a more time-efficient variant of XferBench (which we term XferBench-da).

B Representation of Signalling Game Observations

The originally intended representation for observations in the signalling game was to concatenate one-hot vectors each of which represented the value of of one attribute. For example, the 2-attribute, 3-value vector $[1, 2]$, would be represented as $[0, 1, 0, 0, 0, 1]$ with the first three entries corresponding to the first attribute the last three corresponding to the second attribute. Due to a mistake in the implementation, the actual representation used was simply a vector of the raw integer values such that $[1, 2]$ was simply represented as $[1, 2]$. This to say instead of observations being elements of $\{0, 1\}^{|A|\cdot|V|}$ as originally intended, they were implemented as elements of $\mathbb{Z}^{|A|}$. The agents did not seem to struggle playing this signalling game even with higher numbers of values.

C Hyperparameters Not Discussed

In this section we briefly discuss hyperparameters that were tried but not not documented in the paper or that were not investigated at all. We selected a batch size of 32 based on comparing the compute efficiency of different sizes. Larger batch sizes could process more data faster but would not update the parameters often enough. On the other hand, smaller batch sizes would not process enough data to maximize the utility of each update. Mixed precision training was tested but not found to improve runtime. For learning rate scheduling, we found cosine annealing to be slightly more effective than no schedule, but further schedules were not investigated. Weight decay was investigated in earlier experiment but found not to have a noticeable effect.

The implementation of the signalling game we used could also be optimized using REINFORCE to handle the discrete message, but we only tested with a Gumbel-Softmax layer as it is faster and more stable to optimize with.

D Full Table of Hyperparameters

In Table 3, we show all of the hyperparameters selected for the searches and trials referenced in the paper.

E Computing Resources Used

Experiments were performed across about 20–30 NVIDIA A6000 (or equivalent) GPUs (one trial per GPU) on an institutional cluster. We estimate approximately 5500 GPU-hours were used for all experiments directly related to this paper, including those not documented or directly referenced. The primary searches for the best-performing emergent languages on XferBench (Searches 1–4) took about 1300 GPU-hours.

F Hindi's Outlying Score on XferBench

Both in this paper as well as the original XferBench paper (Boldt and Mortensen, 2024b), Hindi appears to be an outlier in terms of XferBench performance compared to other human languages. *A priori*, we do not have any reason to expect this, especially since the embeddings (and hence lexical information) are not transferred from training to tuning in XferBench. *A posteriori*, we can see that based on the entropy of the Hindi corpus from the XferBench baselines, Hindi's poor performance is *not* an outlier as it follows the trend depicted in Fig. 1 (it is the cluster of green square points along the bottom of the cluster of blue circular points); that is, Hindi's entropy of $\sim$7 bits is unexpectedly low for human language (cf. $\sim$11 bits), but given this low entropy, it performs as expected on XferBench.

While a general data quality program could be causing this low entropy (although Wikipedia data should be relatively clean), we also suspected an encoding problem for Hindi, in part because it is the only baseline language using the particular script (viz. Devanagari, although we do not have a guess why this script would be problematic as compared to others.). Thus, in an informal follow-up experiment, we encoded a parallel text in Mandarin, French, and Hindi (best-, middle-, and worst-performing languages) using both byte-level BPE as well as character-level BPE. The results in Table 4 show that Mandarin is the most efficient for encoding the corpus with byte-level BPE tokens followed by French with Hindi taking more than double the tokens of French. Since XferBench is token-limited, taking more tokens to represent the same data effectively lowers the amount of data that the language model trains on for Hindi, which has a negative effect on downstream performance (i.e., the XferBench score). Using character-level BPE instead yields similar corpus sizes, and, indeed, running XferBench with character-level BPE during training yields similar scores for all three languages (although they have all regressed to Hindi's byte-level BPE performance). Additionally, running XferBench with character-level BPE training led to instabilities with 1 out of 3 runs extremely poor performance, possibly due to character-level BPE being more sensitive to the complete set of characters is the training and tuning corpora.

G Synthetic Languages

G.1 Definitions

We use four probabilistic synthetic languages which span a large portion of the Chomsky hierarchy ranging from trivial to beyond context-free. All synthetic languages contain a unique begin- and end-of-sentence token in each utterance.

Zipf-Mandelbrot Distribution The basis for our synthetic languages will be a Zipf–Mandelbrot distribution, a generalization of Zipf's law, where the unnormalized probability weight of the word w_i is

$$f(w_i) = \frac{1}{(i+\beta)^\alpha}, \tag{1}$$

where i is the 1-based index of the word, α controls the weight of the tail, and β shifts where the distribution starts (roughly speaking). Empirically, $\alpha = 1$ and $\beta = 2.7$ have been found to be good approximations for human language and will be the default parameters of the distribution unless otherwise specified (Piantadosi, 2014).

Bag of Words The simplest synthetic language we introduce is a bag-of-words language where each token in a sentence is sampled independently from the Zipf-Mandelbrot distribution. The length of the sentence is independent of the sampling method, so in interest of simplicity, we sample from a discrete uniform distribution.

Regular The simplest non-trivial language we introduce is a regular language which partitions the tokens uniformly at random into k different sets $(s_1, \ldots, s_k)$, keeping their initial Zipf–Mandelbrot-derived weight. Each sentence starts with a token sampled from s_1; each subsequent token is sampled from the next class $(s_i + 1)$ with probability c or

#	|Trials|	|Attrs.|	|Vals.|	|Distrs.|	Temp.	|Embed.|	|Hidden|	LR	|Vocab|	Length	|Epochs|
1	578	[3, 7]	[3, 7]	[1, 127]	[0.1, 10]	[8, 128]	[8, 128]	[500µ, 50m]	[10, 20k]	[1, 40]	500
2	171	[5, 10]	[5, 10]	—	[0.5, 4]	[64, 512]	[64, 512]	[500µ, 5m]	[300, 30k]	—	—
3	140	—	—	—	—	—	—	—	—	—	[500, 5k]
4	282	[6, 20]	6	23	2	128	256	[1m, 3m]	[500, 30k]	—	—
4.1	1	11	6	—	—	—	—	1.79m	9721	16	1715
4.2	1	12	6	—	—	—	—	1.86m	12496	22	1593
4.3	1	13	6	—	—	—	—	1.74m	8096	18	1511
5r	411	[4, 20]	[3, 10]	[1, 127]	[0.1, 10]	[8, 512]	[8, 512]	[500µ, 10m]	[2, 30k]	[1, 40]	[10, 3k]
6e	109	10	10	[63, 511]	2	32	32	2.7m	25k	15	5k
6e.1	1	—	—	228	—	—	—	—	—	—	—
6e.2	1	—	—	372	—	—	—	—	—	—	—
6e.2	1	—	—	165	—	—	—	—	—	—	—

Table 3: All hyperparameters were treated as log-scale hyperparameters. $|\cdot|$ refers to cardinality. "—" means unchanged from the previous run. µ, m, and k refer to the SI prefixes micro ($\times 10^{-6}$), milli ($\times 10^{-3}$), and kilo ($\times 10^{3}$), respectively. 4.1 is the best-performing trial of Search 4 (and likewise for 4.2, 6e.1, etc.).

Lang	Byte BPETs	Char BPETs	Char XB
zh	470 k	950 k	5.95
fr	900 k	900 k	5.95
hi	2100 k	840 k	5.94

Table 4: BPE token counts for a parallel corpus in various languages and encoding methods. XferBench score with character-level BPE training corpus also provided.

sampled from the same class (s_i). After s_k, the sentence terminates. Thus, the language is defined by the regular expression

$$s_1^+ s_2^+ \dots s_k^+, \tag{2}$$

where $a^+ = aa^*$, s_i represents any token in the set s_i, and appropriate BoS and EoS tokens are added.

Dyck-*n* Dyck-n can be thought of as "balanced nested delimiters" (where the delimiters are the same token) (Schützenberger, 1963). Each token in the sentence is generated as follows: With probability p, a new token is sampled from the Zipf–Mandelbrot distribution and pushed onto a stack (the "opening delimiter"), and with probability $1-p$, the token on top of the stack is popped off. A sentence always begins with an "open" token and ends when the stack is empty. An example of such a sentence is $(3, 1, 1, 2, 1, 1, 2, 3)$ which could be illustrated as "{()[()]}".

Shuffle Dyck-*n* Finally, we use Shuffle Dyck-n as our last language which lies beyond context-free in the Chomsky hierarchy Suzgun et al. (2019). Technically speaking, this language should be called Shuffle of n Distinct Dyck-1 Languages since it is the result of randomly interleaving multiple Dyck-1 languages with distinct tokens. To generate a sentence in Shuffle Dyck-n, we first follow the same procedure as for Dyck-n but keep the individual tokens separate. We then interleave the separate strings by appending to the sentence uniformly at random from one of the individual strings until they are empty. For example, if Dyck-n generated "{([()])[]}", the separated strings would be "{}", "(())", and "[][]", which could then be interleaved into "{[](())]".

G.2 Hyperparameters

Each variation of the synthetic language maintains the default values while varying a single hyperparameter. We vary the common hyperparameters as follows:

Vocabulary size takes the values 10, 100, 1k, 5k, 10k, 30k (default: 30k). A vocab size of 10 is incompatible with the Regular language and was skipped.

Zipf–Mandelbrot α takes the values 0, 0.25, 0.5, 1, 2, and 4 (default: 1).

***n* tokens** (in the whole corpus) takes the values 1k, 10k, 100k, 1M, 5M, and 15M (default: 15M); this hyperparameter was not varied for the Unigram language.

The Unigram language has an additional hyperparameter stop probability which takes the values 0.05, 0.1, and 0.2 (default: 0.1). The Regular language has two additional hyperparameters: repeat probability (c) which takes the values 0.2, 0.4, 0.5, and 0.6 (default: 0.4), and n classes which takes the values 5, 10, 20, and 40 (default: 10). The Dyck and Shuffle Dyck languages take the additional hyperparameter open probability with values:

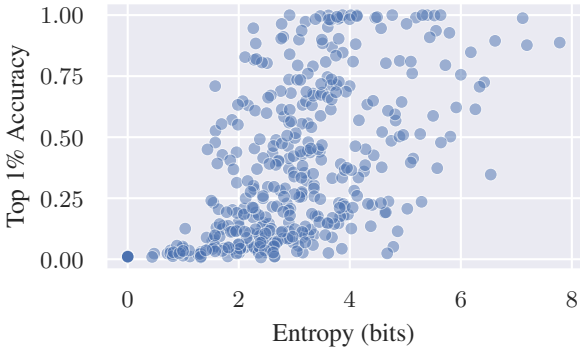

Figure 6: Entropy versus accuracy for Search 5r.

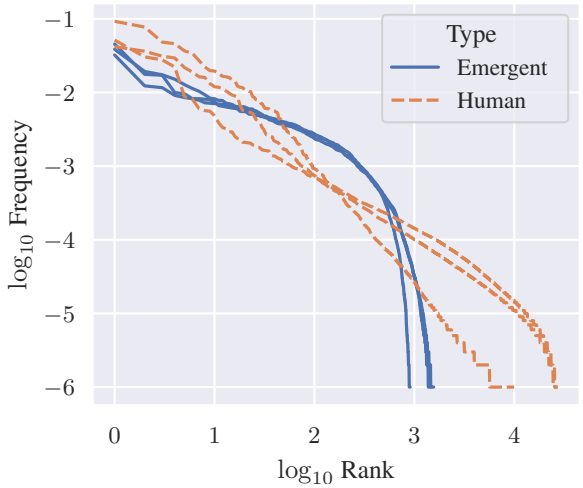

Figure 7: Token log rank versus log frequency plots for emergent and human languages. Logarithms are in base 10.

0.2, 0.3, 0.4, 0.5, and 0.6 (default: 0.5); Shuffle Dyck is not generated with the value 0.6 due to implementation constraints.

H Task Success and Entropy

Previous work (Kharitonov et al., 2020; Chaabouni et al., 2021) has analyzed entropy minimization with respect to the amount of information or, roughly speaking, task success. We performed a brief analysis the relationship between entropy and accuracy (task success) shown in Fig. 6. While we do find significant correlation (Pearson's $r = 0.57$ for Search 5r), we would not characterize it as any strict sort of entropy minimization. That is, we observe many emergent languages which are from the Pareto frontier of high accuracy and low entropy. Hyperparameter search demonstrates itself to be a powerful tool for investigating such correlations since it is able to generate a wide variety of emergent languages with minimal additional work from the researchers. Nevertheless, more investigation would have to be done on this front to conclusively support or reject prior claims of entropy minimization.

I Rank–Frequency Plots

Figure 7 shows Zipf's Law–style plots of rank versus frequency on a log–log scaled plot (Zipf, 1949) for human languages and high-performing emergent languages. As Zipf's Law predicts, the human languages show a roughly linear relationship in log–log space. On the other hand, the emergent languages exhibit more of a "cliff" where higher-ranked tokens have a more similar frequency before quickly falling to near-zero frequency. This implies that human language displays a long tail which is not present in the emergent languages. The fact that the emergent languages studied exhibit this behavior is somewhat expected as the underlying data distribution that they are representing is itself uniform.

J Hyperparameter Scatter Plots

Figures 8 to 11 show the univariate scatter plots for hyperparameter Searches 1–4. The y-axis is XferBench-da score (or some smaller variant thereof, for Searches 1 and 2), and the x-axis is one of the hyperparameters varied for that search. Note that other variables are *not* held constant while one is varied; instead all hyperparameters are varied for each trial.

K Varying Neural Architecture

In a follow-up experiment we test different neural architectures for the sender and receiver agents. In particular, we test different numbers of fully connected layers ($\{1, \ldots, 5\}$), RNN layers ($\{1, \ldots, 5\}$), and RNN types (GRU, LSTM, Elman) (Elman, 1990; Hochreiter and Schmidhuber, 1997; Cho et al., 2014). The number of epochs was also allowed to vary in the event that increasing the number of parameters benefited from longer training. Figure 12 displays the results of this experiment.

The fully connected layers (which surround the sender's and receiver's RNN) have the same hidden size and are separated by tanh activations. The RNN layers vary according to a standard stacked architecture. The *RNN* cell type refers to a plain Elman RNN. The small variant of XferBench-da was used for the objective.

From Fig. 12, we see that LSTMs outperform

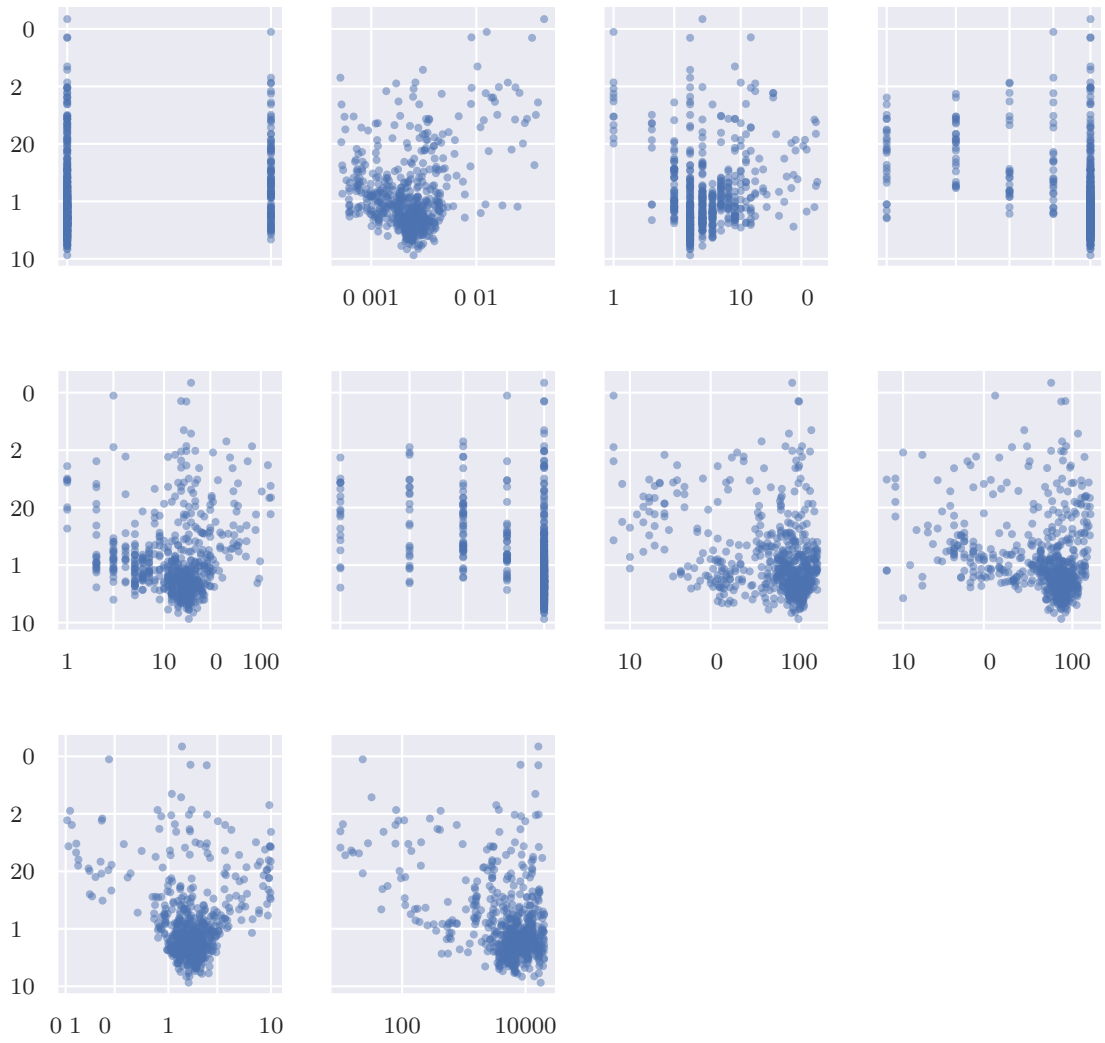

Figure 8: Objective values for Search 1 by individual hyperparameter.

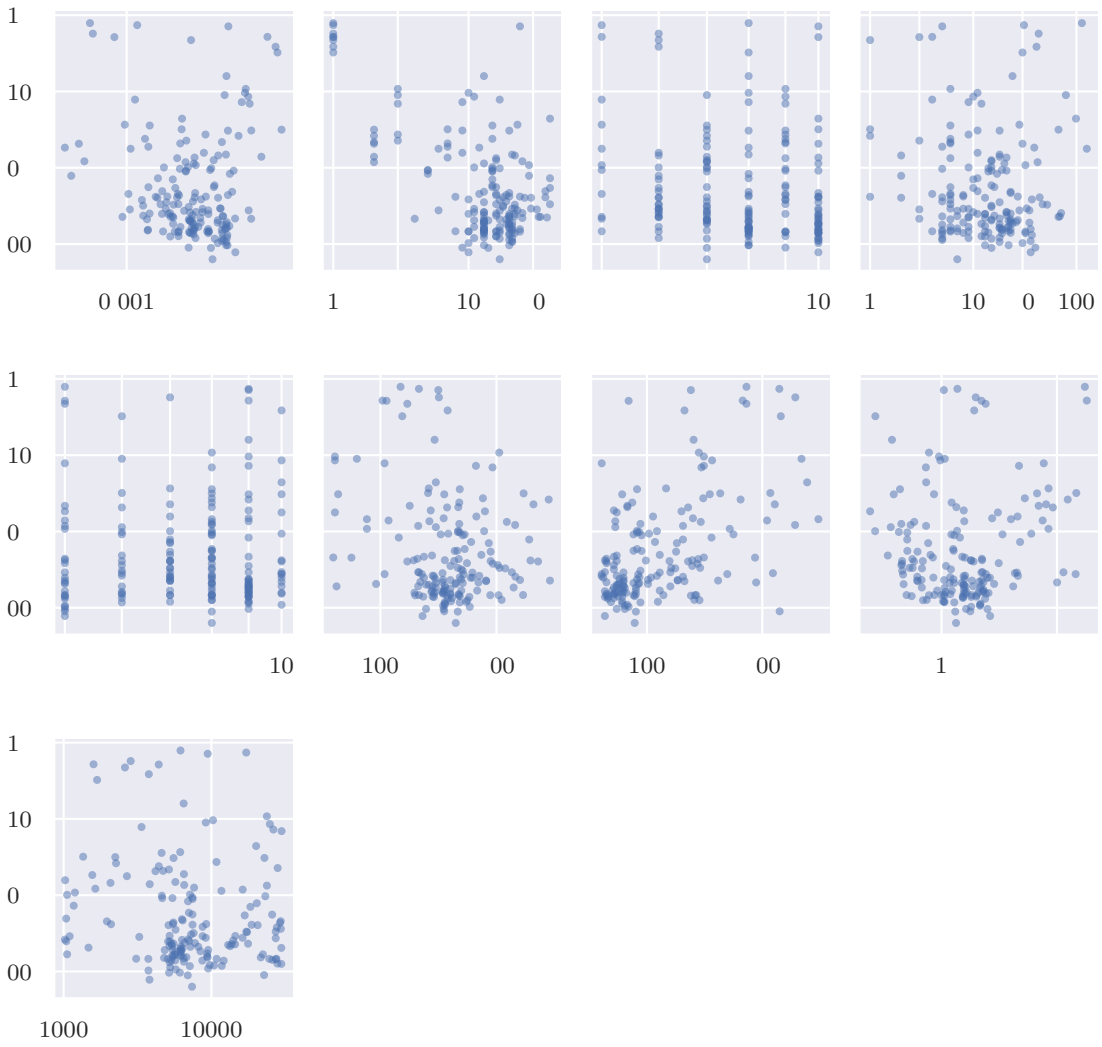

Figure 9: Objective values for Search 2 by individual hyperparameter.

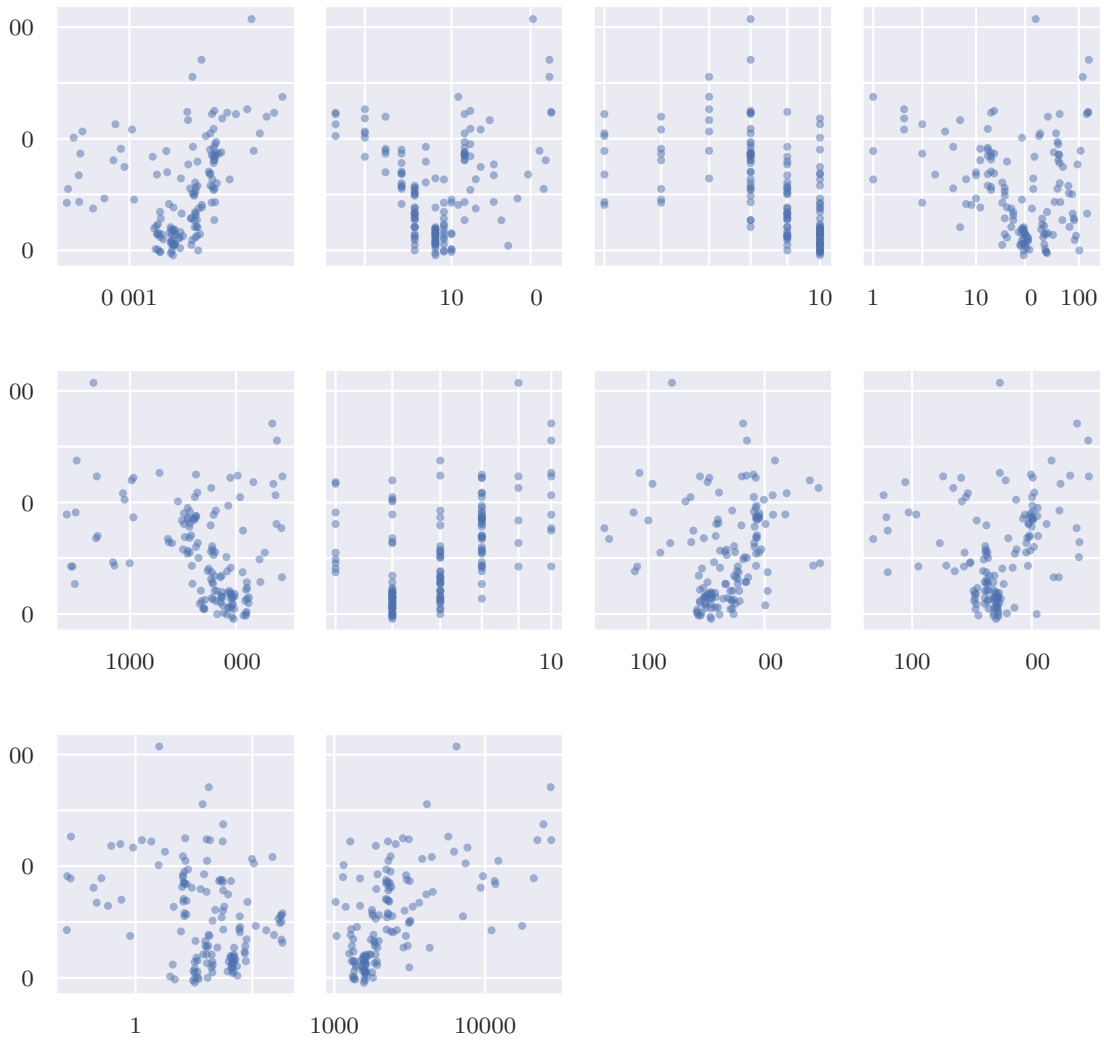

Figure 10: Objective values for Search 3 by individual hyperparameter.

Figure 11: Objective values for Search 4 by individual hyperparameter.

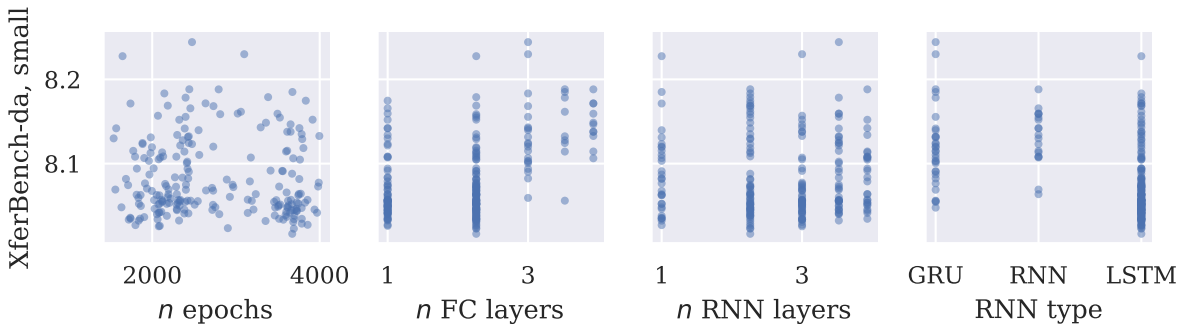

Figure 12: Objective values for Search 7 by individual hyperparameter.

GRUs (used for the main experiments) and RNNs by a large margin. On the other hand, Using 2 instead of 1 layer (used for the main experiments) provides a smaller performance gain on XferBench-d while further increasing the layers does not show improvement. The number of epochs did not have a notable effect.