Principled Evaluation of Deep Learning-Based Emergent Communication

Brendon J. Boldt

February 3, 2026

Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee

David Mortensen, *Chair*	Carnegie Mellon University
Yonatan Bisk	Carnegie Mellon University
Daniel Fried	Carnegie Mellon University
Kenny Smith	University of Edinburgh

February 3, 2026

Thesis Statement

Advancing science and engineering with deep learning-based emergent language requires principled, environment-agnostic analytical algorithms.

Abstract

Emergent communication is the field of research which studies how human language-like communication systems evolve from scratch in agent-based simulations. The most recent incarnation of this topic, starting in 2016, has focused on leveraging recent advancements in deep neural network, reinforcement learning, and natural language processing. Emergent communication, as a method, has significant potential applications from powering unprecedentedly detailed simulations of how humans invent, acquire, and use language to providing an alternative to extracting language data from humans to train large language models. Despite this potential, the field has yet to make progress towards the more revolutionary applications because it lacks the methodological resources to enable cumulative research; that is, research findings are often "one-off", lacking ways to build on prior or make generalizable claims.

This thesis, then, advances the field of emergent communication by developing the resources necessary for a cumulative research paradigm, something that is critical to the advancement of any area of science or engineering. Specifically, it establishes methods in emergent communication that enable measurable progress in emergent language research so as to move the field towards solving practical applications and improving scientific understanding. It does this by first introducing emergent language data resources which enable empirical evaluation across a variety of emergent languages. These resources are then used to develop (1) a deep transfer learning-based evaluation metric for emergent communication to measure the practical applicability of emergent language and (2) algorithms for discovering the morphology of emergent languages as a foundation for further linguistic analysis.

Contents

1 Introduction **4**
- 1.1 Background . . . 5
- 1.2 Motivation . . . 7
- 1.3 Overview . . . 7

2 Building a Library of Emergent Languages **11**
- 2.1 Introduction . . . 11
- 2.2 Related Work . . . 12
- 2.3 Design . . . 13
- 2.4 Content . . . 15
- 2.5 Analysis . . . 19
- 2.6 Discussion . . . 22
- 2.7 Conclusion . . . 24

3 Adding Semantic Annotations to Corpora [proposed] **25**
- 3.1 Introduction . . . 25
- 3.2 Format . . . 26
- 3.3 Experiments . . . 29

4 Evaluation of Emergent Languages with Deep Transfer Learning **31**
- 4.1 Introduction . . . 31
- 4.2 Related Work . . . 33
- 4.3 XferBench . . . 34
- 4.4 Experiments . . . 37
- 4.5 Results . . . 41
- 4.6 Discussion . . . 43
- 4.7 Conclusion . . . 45
- 4.8 Limitations . . . 45

5 Optimizing for Human Language Similarity with Transfer Learning [under review] **46**
- 5.1 Introduction . . . 46
- 5.2 Related Work . . . 48
- 5.3 Methods . . . 48
- 5.4 Experiments . . . 51

5.5 Analysis . . . 54
5.6 Discussion . . . 56
5.7 Conclusion . . . 57

6 Discovering Morphemes in Rich Corpora [proposed] 59
6.1 Introduction . . . 59
6.2 Problem Definition . . . 60
6.3 Algorithm . . . 62
6.4 Empirical Validation . . . 64
6.5 Analysis of Emergent Languages . . . 69
6.6 Discussion . . . 72
6.7 Conclusion . . . 72
6.8 Limitations . . . 73

7 Detecting Structure Among Morphemes [proposed] 74
7.1 Introduction . . . 74
7.2 Algorithm . . . 75
7.3 Experiments . . . 79

8 Conclusion 80

Bibliography 81

A ELCC 96

B XferBench 103

C Optimizing with Transfer Learning 111

D Morpheme Induction 120

Chapter 1

Introduction

Modern-day large language model-based AI systems are good a mimicking human language. Some might even say they are good at *using* human language, but this is either imprecise or inaccurate: LLMs' production of text is based the statistical likelihood of meaningless (so far as they are concerned) tokens, fine-tuned to humans' preferences. This is in contrast to humans' use—and event more so their acquisition—of language which is laden with meaning derived from the rich internal, physical, and social context which permeates language use. The end result is that LLMs' approximation of human language fails at a number of tasks, but, more significantly, falls short in providing insights into the nature of human language itself.

Emergent communication (also known as *emergent language*) is an alternative paradigm to developing language-capable models that does not does not train on human language data but rather invents a communication system *de novo*. In its most basic form, emergent communication comprises a simulation using neural network-based agents which are trained to cooperatively complete some task in a virtual environment. These agents are equipped with a communication channel of discrete tokens with no *a priori* meaning—the meaning of communication is established through the optimization process encouraging communication which is advantageous to completing the task.

Emergent communication differs from the more "traditional" approach to language that LLMs use in that it does not try mimic human language but instead tries to rederive language from similar function pressures which are hypothesized to have guided human language's own evolution. Since the process language emerging is far more analogous to how human language develops and is learned, it has a much greater potential to yield significant gains the scientific understanding of human language. Furthermore, certain practical tasks might lie beyond the reach of the mimicry approach LLMs employ due to surface-level operations; these problems, too, can be addressed by emergent communication which models not only the surface features of language but also its semantics and social context. Finally, generating language data through emergent communication avoids many of the ethical issues that crop up with LLMs dependence on human language data from amplifying toxicity from the Web to freely (ab)using copyrighted and personal content [Weidinger et al., 2021, Carlini et al., 2021].

Here is the problem: the field of emergent communication has not made measurable progress toward its more revolutionary applications to scientific understanding or practical

problems. This thesis, then, establishes methods in emergent communication that enable measurable progress in emergent language research, enabling practitioners to tackle the most significant applications of the field through cumulative contributions. It does this by first introducing emergent language data resources which permit empirical evaluation across a variety of emergent languages. These resources are then used to develop environment-agnostic analytical algorithms centering on the key applications of emergent communication. Namely, this thesis introduces (1) a deep transfer learning-based evaluation metric for emergent communication to measure the practical applicability of emergent language and (2) algorithms for discovering morphology of emergent languages as a foundation for further linguistic analysis.

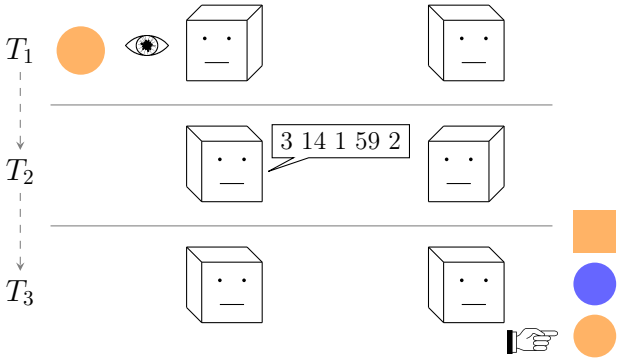
1.1 Background

The field of deep learning-based emergent communication has its genesis in 2016 with papers including "Learning to communicate with deep multi-agent reinforcement learning" [Foerster et al., 2016] and "Multi-Agent Cooperation and the Emergence of (Natural) Language" [Lazaridou et al., 2016]. These were the first paper to combine combine deep learning, and specifically deep reinforcement learning, to developing discrete token-based communication systems from scratch. While prior work applied mathematical models [Brighton et al., 2005] and classical machine learning methods [Werner and Dyer, 1991], the introduction of deep learning opened up the possibility of a far more robust notion of the results of the simulations being *emergent*. That is, with mathematical models the range of results is tightly constrained by the design of the model and "emergent" phenomena are either relatively simple or encoded into the model itself.[1] Deep reinforcement learning, on the other hand, has demonstrated vivid example of complex behaviors emerging in environments with simple rules such as DeepMind's AlphaZero [Silver et al., 2017] or OpenAI's multi-agent hide-and-seek [Baker et al., 2020].

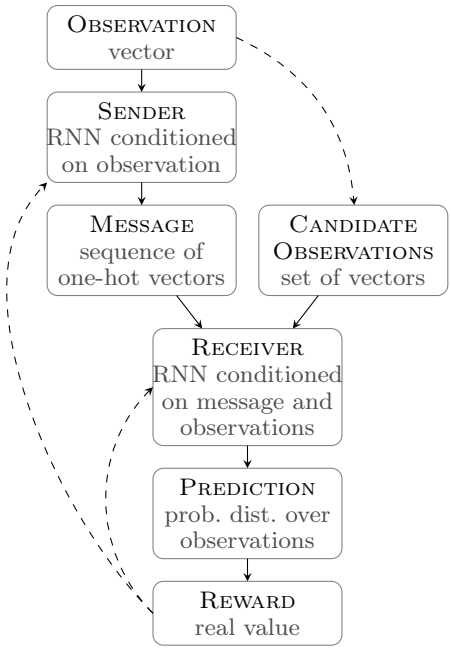
The prototypical emergent communication experiment consists two or more deep neural network-based agents situated in some kind of environment or game where they must cooperate in order to succeed. The agents are equipped with a communication channel consisting of discrete tokens with no *a priori* meaning; it is only through the reinforcement learning-based optimization that messages passed between agents begin to take on meaning. The resulting behavior, most especially the communication protocol, is the typically the object analysis, addressing question such as: Did an effective communication protocol emerge at all? What structural features characterize it? Do these features align at all to human language? What can we infer about language formation more generally from the above?

In practice, much of the literature has focused on the signalling game and the emergence of compositionality in communication (jointly and separately) [Havrylov and Titov, 2017, Mordatch and Abbeel, 2018, Chaabouni et al., 2022]. The signalling game was introduced in the context of game theory by David Lewis [Lewis, 1969]. The signalling game is one of the simplest possible environments for emergent communication contributing, in large part, to its popularity. It consists of only two agents: a sender and a receiver. The sender makes on observation (e.g., an orange circle) and sends a message to the receiver who must,

[1] Although Conway's Game of Life is notable exception to this.



(a) T_1: The sender (left) observes an object. T_2: The sender passes a message to the receiver (right). T_3: The receiver chooses from a handful of candidate objects.



(b) Illustration of the technical architecture of the signaling game.

Figure 1.1: An illustration of the discrimination variant of the signaling game, one of the simplest and most common environments in emergent communication research.

based on the message alone, determine the nature of the observation (e.g., it was an orange circle, not a blue circle). A visualization of the signalling game is provided in Figure 1.1. The question of compositionality arises when we look at how the communication protocol encodes compound meanings like a red square: A compositional protocol would encode "red" and "circle" with their own words which could be reused to express meanings like a red circle or a blue square. On the other hand, holistic communication sometimes emerges where a unique word refers to red square, bearing no relation to the word(s) for red circle. Compositional communication, generally, is seen as more desirable both for practical reasons (more efficient encoding of information) as well as for its resemblance to how humans tend to encode meaning in language.

Other environments do appear in the literature such as navigation tasks or dialogue-based games [Unger and Bruni, 2020, Brandizzi et al., 2022]. In addition to compositionality, other phenomena have been the subject of investigation such as pragmatics, transfer learning, and the information theoretic properties of emergent language [Kang et al., 2020, Yao et al., 2022, Tucker et al., 2021]. While there are too many papers to summarize comprehensively here, I estimate that there are on the order of 200 papers directly related to emergent communication.[2] For a general review of the emergent communication literature, I recommend Lazaridou and Baroni [2020].

[2]Figure based on finding ∼150 papers during a comprehensive literature review in 2023.

1.2 Motivation

In the seven or so years of deep learning-based emergent communication's existence, there has been little measurable progress toward the farther-reaching applications of the field. Much of the research has focused on addressing small-scale, isolated phenomena without a clear way to unify the findings into a broader understanding. While normal science (as Kuhn terms it [Kuhn, 1962]) often proceeds by small, additive research contributions, emergent communication has not developed paradigm where the small contributions can truly add together. One can read much of the literature on emergent communication, learn of many different trends that appeared in particular environments, and still largely have little idea why emergent languages look the way they do nor what they might look like in a new environment. Furthermore, despite the potential for groundbreaking applications in natural language processing and linguistics, emergent communication has not yielded any broadly-applicable contributions either [Boldt and Mortensen, 2024c].

This thesis is intended to make foundational contributions addressing the unique structural issues in emergent communication research that has resulted in this. If the same old, tried and true methods of machine learning simply worked for emergent language, we would have seen tangible progress by now, but we have not, so one can assume that emergent language is a dead end, is missing a key technological prerequisite (e.g., computing resources, better reinforcement learning algorithms), or needs specially tailored methodological improvements. This thesis is an attempt at the last of these. In particular, the overall intent of the thesis is to create quantitative evaluative methods which work across a wide variety of emergent communication environments in order to support a research and development workflow more like that of the rest of deep learning-based research. Methods should yield *quantitative* metrics to permit direct comparison of different systems, statistical analysis, and more automated methods of exploring emergent communication environments. These methods must also be *evaluative* since we a number that we can optimize for—a goal—not simply a number describing one aspect of a system that requires further interpretation. Finally, this thesis looks to rest of deep learning research, especially in regards to structures like benchmarks and evaluation metrics, for inspiration because these factors are critical for its own success. While the long-term development of emergent communication methods needs far more than just borrowing methods from deep learning, developing general-purpose evaluative tools is critical in unifying the research efforts of the field such that they can begin to progress in a tangible way.

1.3 Overview

This thesis has three main parts: The first comprises Chapters 2 and 3, which introduce a large collection of emergent language corpora and corresponding semantic annotations. Second, Chapters 4 and 5 introduce and showcase XferBench, a deep transfer learning-based evaluation metric for emergent language corpora. Lastly, Chapters 6 and 7 introduce methods for detecting linguistic structures in emergent language corpora. Each of these parts largely stands on its own for the purposes of readability as well as their contributions. Nevertheless, the entire thesis is intended to provide a coherent, multi-pronged approach to

making measurable progress in emergent language possible.

Data resources for emergent language

Chapters 2 and 3 introduce an important data resource to emergent communication research: the Emergent Language Corpus Collection (ELCC), a collection of emergent language corpora with semantic annotations of utterances derived from a variety of free and open source emergent communication implementations. Each of these corpora is accompanied by metadata describing statistical properties of the corpus, taxonomic properties of the environment it came from, and a turnkey shell script for reproducing the corpus (or developing a new one). This collection of corpora is made a public such that it can be both easily used and contributed to by the broader research community.

In its own right, this collection is a significant contribution to emergent communication as it increases the accessibility of emergent language data both for researchers who might be able to generate the emergent languages themselves and those looking to compare a wide variety of emergent languages. More importantly for the thesis, though, having a robust collection of emergent language corpora is necessary test, contextualize, and motivate the results of the following chapters. The transfer learning-based metric discussed below takes emergent language corpora as input, and to demonstrate its utility in comparing various approaches to emergent communication, it needs to be applied to a wide variety of emergent languages. For the methods of detecting linguistic structure, not only is the variety of emergent languages important, but the semantic annotations are a critical part of discovering the latent structure that might be present in the emergent languages. Thus, the collection of emergent language corpora forms the foundation for the rest of the research in this thesis while also demonstrating how emergent communication research can be made more accessible.

Transfer learning-based evaluation

Chapters 4 and 5 introduce XferBench, an evaluation metric for emergent language corpora based on deep transfer learning. The intuition behind XferBench is that when a model is pretrained on emergent language data, its downstream performance on human language-based natural language processing tasks (e.g., language modelling, machine translation) is correlated with its similarity to human language, as far as a deep neural network is concerned. XferBench is packaged as a benchmark: standardized data and settings with a clean, easy-to-use implementation to permit widespread.

XferBench exemplifies the goal of the thesis insofar as it establishes an evaluative method for easily and effectively comparing emergent languages on a level playing field. The transfer learning-based approach captures notion of how "good" an emergent language (corpus) is from the perspective of machine learning. This is meant in two ways: First, the emergent languages are analyzed according to the methods of machine learning, discovering regularly occurring patterns with data-driven methods and a low inductive bias. Second, transfer learning method closely mirrors many of the practical applications of emergent communication which consist of using emergent language data is pretraining or evaluation data for natural language processing models. While it is not as simple to establish a ranking of "better" and "worse" with the largely open ended task of designing an emergent communication environment,

XferBench still provides a useful notion of what directions are having a tangible effect on the complexity of the emergent languages those environments develop.

Detecting linguistic structures

Chapters 6 and 7 introduce algorithms for detecting linguistic structures in any emergent language corpora that possesses in annotations as described above in ELCC. Chapter 6 specifically looks at detecting "morphemes" in the sense of atomic units of form with a distinct meaning. The output is a list of token sequences which correspond with particular meanings in the environment. Not only does this enable a host of interesting analyses, but it also lays the groundwork for Chapter 7 which introduces an algorithm to detect structure among these morphemes, making a first step towards identifying the syntax of emergent languages.

The original intent for the linguistics-focused component of this thesis was to develop a benchmark similar in intent to XferBench but looking at how close the linguistic features of emergent language were to those of human language in areas such as syntax, social variation, and discourse. Yet upon planning the concrete details of such a metric, it was apparent that it was not clear *if* such features as syntax or discourse existed in any meaningful way, let alone there being a way to detect them. Thus, in order to help decide where to begin we visualized and informal hierarchy of linguistic phenomena in Figure 1.2; the direction of the hierarchy is determined by what phenomena presuppose the existence of more basic phenomena. What was immediately apparent is that while it easy to point to concrete notions of semantics and tokens (the atomic components of a message/utterance in emergent communication), even even the existence and nature of the immediate combination of these—morphemes—was not established (and hardly explored).

Thus, I decided to pursue establishing a method to show the existence of and identify morphemes and syntax—the backbone of the hierarchy depicted in Figure 1.2, which is likely the most that can be done toward developing a metric of linguistic similarity in the scope of this thesis. Nevertheless, the proposed algorithms still fit well within this thesis' theme of pioneering accessible general purpose in methods in emergent communication that permit the direct comparison of emergent languages across a wide of environment and implementations.

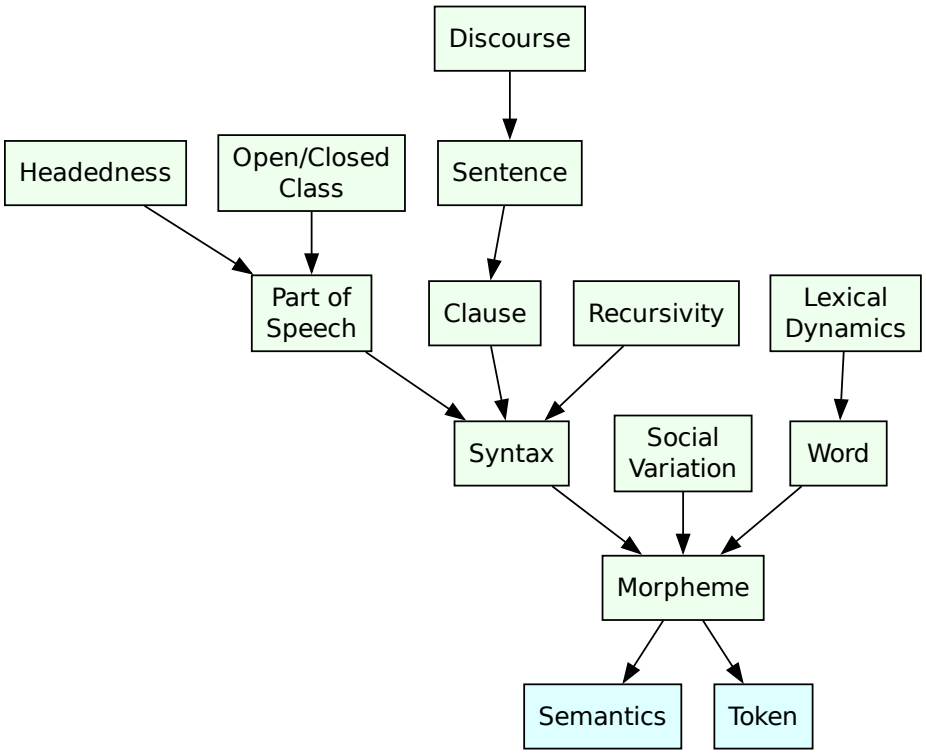


Figure 1.2: Hierarchy of linguistic concepts. $X \to Y$ can be read as "the definition of X presupposes Y being defined" or roughly "X depends on Y". The only concepts whose existence is established in emergent language are *semantics* and *token*.

Chapter 2

Building a Library of Emergent Languages

Abstract

This chapter introduces the Emergent Language Corpus Collection (ELCC): a collection of corpora generated from open source implementations of emergent communication systems across the literature. These systems include a variety of signalling game environments as well as more complex environments like a social deduction game and embodied navigation. Each corpus is annotated with metadata describing the characteristics of the source system as well as a suite of analyses of the corpus (e.g., size, entropy, average message length, performance as transfer learning data). Currently, research studying emergent languages requires directly running different systems which takes time away from actual analyses of such languages, makes studies which compare diverse emergent languages rare, and presents a barrier to entry for researchers without a background in deep learning. The availability of a substantial collection of well-documented emergent language corpora, then, will enable research which can analyze a wider variety of emergent languages, which more effectively uncovers general principles in emergent communication rather than artifacts of particular environments. We provide some quantitative and qualitative analyses with ELCC to demonstrate potential use cases of the resource in this vein.[1]

2.1 Introduction

When Boldt and Mortensen [2024b] introduced the metric called XferBench, they raised a question that they apparently could not answer: how do emergent languages—communication systems that emerge from scratch in agent-based simulations—differ in their "humanlikeness" (as measured by their utility as pretraining data for NLP tasks). It seems likely that they were unable to answer this question because no representative collection of samples from emergent languages existed. The same problem plagues other research programs that seek to make generalizations about emergent languages, as a whole, rather than using a single

[1] Based on "ELCC: the Emergent Language Corpora Collection" available on arXiv [Boldt and Mortensen, 2024a].

type of environment as a proof of concept. These include the degree to which emergent languages display entropic patterns similar to those that characterize words in human languages [Ueda et al., 2023] and the kind of syntax that can be detected in emergent languages through grammar induction [van der Wal et al., 2020]. We present an initial solution to this problem, namely the Emergent Language Corpus Collection (ELCC): a collection of 73 corpora generated from 7 representative emergent communication systems (ECSs).[2] Prior to this work, comparing emergent languages entailed extensive work getting free and open source simulations to run—managing dependencies, manipulating output formats, etc.—before any data could even be generated. The current work allows investigators, even those with very limited software engineering knowledge, to analyze a wide range of emergent languages straightforwardly, plowing over a barrier that has held back comparative emergent language research from its inception. ELCC is published at `https://huggingface.co/datasets/bboldt/elcc` with data and code licensed under the CC BY 4.0 and MIT licenses, respectively.

We discuss related work in Section 2.2. Section 2.3 lays out the design of ELCC while Section 2.4 describes the content of the collection. Section 2.5 demonstrates some of the types of analyses enabled by ELCC. Section 2.6 presents some brief analyses, discussion, and future work related to ELCC. Finally, we conclude in Section 2.7.

Contributions The primary contribution of this paper is as a first-of-its kind data resource which will enable broader engagement and new research directions within the field of emergent communication. Additionally, code published for reproducing the data resource also improve the reproducibility of existing ECS implementations in the literature, supporting further research beyond just the data resource itself. Finally, the paper demonstrates some of the analyses uniquely made possible by a resource such as ELCC.

2.2 Related Work

Emergent communication There is no direct precedent for this work in the emergent communication literature that we are aware of. Perkins [2021b] introduces the TexRel dataset, but this is a dataset of observations for training ECSs, not data generated by them. Some papers do provide the emergent language corpora generated from their experiments (e.g., Yao et al. [2022]), although these papers are few in number and only include the particular ECS used in the paper. At a high level, the EGG framework [Kharitonov et al., 2019] strives to make emergent languages easily accessible, though instead of providing corpora directly, it provides a framework for implementing ECSs. Thus, while EGG is useful for someone building new systems entirely, it is not geared towards research projects aiming directly at analyzing emergent languages themselves.

[2] Emergent communications systems are more commonly referred to as simply "environments"; we choose to use the term "system" in order to emphasize that what goes into producing an emergent language is more than just an environment including also the architecture of the agents, optimization procedure, datasets, and more.

```
systems/ ..... top-level directory
  ecs-1/ ..... directory for a particular ECS
    metadta.yml ..... metadata about the ECS
    code/ ..... directory containing files to produce the data
    data/ ..... directory containing corpus and metadata files
      hparams-1/ ..... directory for run with specific hyperparameters
        corpus.jsonl ..... corpus data
        metadata.json ..... metadata specific for corpus (e.g., metrics)
      hparams-2/ ..... as above
      hparams-n/ ..... as above
  ecs-2/ ..... as above
  ecs-n/ ..... as above
```

Figure 2.1: The file structure of ELCC.

Data resources At a high level, ELCC is a collection of datasets, each of which represent a particular instance of a phenomenon (emergent communication, in this case). On a structural level, ELCC is analogous to a collection of different human languages in a multi-lingual dataset. ELCC, though, focuses more on a particular phenomenon of scientific interest, and, in this way, would be more analogous to work such as Blum et al. [2023], which presents a collection of grammar snapshot pairs for 52 different languages as instances of diachronic language change. Similarly, Zheng et al. [2024] present a dataset of conversations from Chatbot Arena, where "text generated by different LLMs" is the phenomenon of interest. Furthermore, insofar as ELCC documents the basic typology of different ECSs, it is similar to the World Atlas of Language Structures (WALS) [Dryer and Haspelmath, 2013].

2.3 Design

Format

ELCC is a collection of ECSs, each of which has one or more associated *variants* which correspond to runs of the system with different hyperparameter settings (e.g., different random seed, message length, dataset). Each variant has metadata along with the corpus generated from its settings. Each ECS has its own metadata as well and code to generate the corpus and metadata of each variant. The file structure of ELCC is illustrated in Figure 2.1.

ECS metadata Environment metadata provides a basic snapshot of a given system and where it falls in the taxonomy of ECSs. As the collection grows, this structure makes it easier to ascertain the contents of the collection and easily find the most relevant corpora for a given purpose. This metadata will also serve as the foundation for future analyses of the corpora by looking at how the characteristics of an ECS relate to the properties of its output. These metadata include:

- Source information including the original repository and paper of the ECS.
- High-level taxonomic information like game type and subtype.

- Characteristics of observation; including natural versus synthetic data, continuous versus discrete observations.
- Characteristics of the agents; including population size, presence of multiple utterances per episode, presence of agents that send *and* receive messages.
- Free-form information specifying the particular variants of the ECS and general notes about the ELCC entry.

A complete description is given in Section A.1. These metadata are stored as YAML files in each ECS directory. A Python script is provided to validate these entries against a schema. See Section A.2 for an example of such a metadata file.

Corpus Each *corpus* comprises a list of *lines* each of which is, itself, an array of *tokens* represented as integers. Each line corresponds to a single episode or round in the particular ECS. In the case of multi-step or multi-agent systems, this might comprise multiple individual utterances which are then concatenated together to form the line (no separation tokens are added). Each corpus is generated from a single run of the ECS; that is, they are never aggregated from distinct runs of the ECS.

Concretely, a *corpus* is formatted as a JSON lines (JSONL) file where each *line* is a JSON array of integer *tokens* (see Figure 2.3 for an example of the format). There are a few advantages of JSONL: (1) it is a human-readable format, (2) it is JSON-based, meaning it is standardized and has wide support across programming languages, and (3) it is line-based, meaning it is easy to process with command line tools.[3] Corpora are also available as single JSON objects (i.e., and array of arrays), accessible via the Croissant ecosystem [Akhtar et al., 2024].

Corpus analysis For each corpus in ELCC we run a suite of analyses to produce a quantitative snapshot. This suite metrics is intended not only to paint a robust a picture of the corpus but also to serve as jumping-off point for future analyses on the corpora. Specifically, we apply the following to each corpus: token count, unique tokens, line count, unique lines, tokens per line, tokens per line stand deviation, 1-gram entropy, normalized 1-gram entropy, entropy per line, 2-gram entropy, 2-gram conditional entropy, EoS token present, and EoS padding. *Normalized* 1*-gram entropy* is computed as 1*-gram entropy* divided by the maximum entropy given the number of unique tokens in that corpus.

We consider an EoS (end-of-sentence) token to be present when: (1) every line ends with token consistent across the entire corpora, and (2) the first occurrence of this token in a line is only ever followed by more of the same token. For example, `0` could be an EoS token in the corpus `[[1,2,0],[1,0,0]]` but not `[[1,2,0],[0,1,0]]`. EoS padding is defined as a corpus having an EoS token, all lines being the same length, and the EoS token occurs more than once in a line at least once in the corpus.

Additionally, each corpus also has a small amount of metadata copied directly from the output of the ECS; for example, this might include the success rate in a signalling game

[3] E.g., Creating a 100-line random sample of a dataset could be done with `shuf dataset.jsonl | head -n 100 > sample.jsonl`

Source	Type	Data source	Multi-agent	Multi-step	n corp.
Kharitonov et al. [2019]	signalling	synthetic	No	No	15
Yao et al. [2022]	signalling	natural	No	No	2
Mu and Goodman [2021b]	signalling	both	No	No	6
Chaabouni et al. [2022]	signalling	natural	Yes	No	5
Unger and Bruni [2020]	navigation	synthetic	No	Yes	18
Boldt and Mortensen [2023]	navigation	synthetic	No	Yes	20
Brandizzi et al. [2022]	conversation	—	Yes	Yes	7

Table 2.1: Taxonomic summary the contents of ELCC.

environment. We do not standardize this because it can vary widely from ECS to ECS, though it can still be useful for comparison to other results among variants within an ECS.

Reproducibility ELCC is designed with reproducibility in mind. With each ECS, code is included to reproduce the corpora and analysis metadata. Not only does this make ELCC reproducible, but it sometimes helps the reproducibility of the underlying implementation insofar as it fixes bugs, specifies Python environments, and provides examples of how to run an experiment with a certain set of hyperparameters. Nevertheless, in this code, we have tried to keep as close to the original implementations as possible. When the underlying implementation supports it, we set the random seed (or keep the default) for the sake of consistency, although many systems do not provide a way to easily set this.

2.4 Content

ELCC contains 73 corpora across 8 ECSs taken from the literature for which free and open source implementations were available. With our selection we sought to capture variation across a three distinct dimensions:

1. Variation across ECSs generally, including elements like game types, message structure, data sources, and implementation details.

2. Variation among different hyperparameter settings within an ECS, including message length, vocabulary size, dataset, and game difficulty.

3. Variation within a particular hyperparameter setting that comes from inherent stochasticity in the system; this is useful for gauging the stability or convergence of an ECS.

Table 2.1 shows an overview of the taxonomy of ELCC based on the ECS-level metadata. In addition to this, Table 2.2 provides a quantitative summary of the corpus-level metrics described in Section 2.3. We separate the discussion of particular systems into two subsections: signalling games (Section 2.4) and its variations which represent a large proportion of system discussed in the literature and other games (Section 2.4) which go beyond the standard signalling framework.

Scope

The scope of the contents of ELCC is largely the same as discussed in reviews such as Lazaridou and Baroni [2020] and Boldt and Mortensen [2024c, Section 1.2]. This comprises agent-based models for simulating the formation of "natural" language from scratch using deep neural networks. Importantly, *from scratch* means that the models are not pretrained or tuned on human language. Typically, such simulations make use of reinforcement learning to train the neural networks, though this is not a requirement in principle.

One criterion that we do use to filter ECSs for inclusion is its suitability for generating corpora as described above. This requires that the communication channel is discrete, analogous to the distinct words/morphemes which for the units of human language. This excludes a small number of emergent communication papers have approached emergent communication through constrained continuous channels like sketching [Mihai and Hare, 2021b] or acoustic-like signals [Eloff et al., 2023]. Other systems use discrete communication but have episodes with only a single, one-token message (e.g., Tucker et al. [2021]), which would have limited applicability to many research questions in emergent communication.

Signalling games

The *signalling game* (or *reference game*) [Lewis, 1969] represents a plurality, if not majority, of the systems present in the literature. A brief, non-exhaustive review of the literature yielded 43 papers which use minor variations of the signalling game, a large number considering the modest body of emergent communication literature (see Section A.3). The basic format of the signalling game is a single round of the *sender* agent making an observation, passing a message to the *receiver* agent, and the receiver performing an action based on the information from the message. The popularity of this game is, in large part, because of its simplicity in both concept and implementation. Experimental variables can be manipulated easily while introducing minimal confounding factors. Furthermore, the implementations can entirely avoid the difficulties of reinforcement learning by treating the sender and receiver agents as a single neural network, resulting in autoencoder with a discrete bottleneck which can be trained with backpropagation and supervised learning.

The two major subtypes of the signalling game are the *discrimination game* and the *reconstruction game*. In the discrimination game, the receiver must answer a multiple-choice question, that is, select the correct observation from among incorrect "distractors". In the reconstruction game, the receiver must recreate the input directly, similar to the decoder of an autoencoder.

Vanilla For the most basic form of the signalling game, which we term "vanilla", we use the implementation provided in the Emergence of lanGuage in Games (EGG) framework [Kharitonov et al., 2019, MIT license]. It is vanilla insofar as it comprises the signalling game with the simplest possible observations (synthetic, concatenated one-hot vectors), a standard agent architecture (i.e., RNNs), and no additional dynamics or variations on the game. Both the discrimination game and the reconstruction game are included. This system provides a good point of comparison for other ECSs which introduce variations on the signalling game. The simplicity of the system additionally makes it easier to vary hyperparameters:

for example, the size of the dataset can be scaled arbitrarily and there is no reliance on pretrained embedding models.

Natural images "Linking emergent and natural languages via corpus transfer" [Yao et al., 2022, MIT license] presents a variant of the signalling game which uses embeddings of natural images as the observations. In particular, the system uses embedded images from the MS-COCO and Conceptual Captions datasets consisting of pictures of everyday scenes. Compared to the uniformly sampled one-hot vectors in the vanilla setting, natural image embeddings are real-valued with a generally smooth probability distribution rather than being binary or categorical. Furthermore, natural data distributions are not uniform and instead have concentrations of probability mass on particular elements; this non-uniform distribution is associated with various features of human language (e.g., human languages' bias towards describing warm colors [Gibson et al., 2017, Zaslavsky et al., 2019]).

Concept-based observations "Emergent communication of generalizations" [Mu and Goodman, 2021b, MIT license] presents a variant of the discrimination signalling game which they term the *concept game*. The concept game changes the way that the sender's observation corresponds with the receiver's observations. In the vanilla discrimination game, the observation the sender sees is exactly the same as the correct observation that the receiver sees. In the concept game, the sender instead observes a set of inputs which share a particular concept (e.g., red triangle and red circle are both red), and the correct observation (among distractors) shown to the receiver contains the same concept (i.e., red) while not being identical to those observed by the sender. The rationale for this system is that the differing observations will encourage the sender to communicate about abstract concepts rather than low-level details about the observation. This ECS also presents the vanilla discrimination game as well as the *set reference game*, which is similar to the reference game except that the whole object is consistent (e.g., different sizes and locations of a red triangle).

Multi-agent population "Emergent communication at scale" [Chaabouni et al., 2022, Apache 2.0-license] presents a signalling game system with populations of agents instead of the standard fixed pair of sender and receiver. For each round of the game, then, a random sender is paired with a random receiver. This adds a degree of realism to the system, as natural human languages are developed within a population and not just between two speakers (cf. idioglossia). More specifically, language developing among a population of agents prevents some degree "overfitting" between sender and receiver; in this context, having a population of agents functions as an ensembling approach to regularization.

Other games

Considering that the signalling game is close to the simplest possible game for an ECS, moving beyond the signalling game generally entails an increase in complexity. There is no limit to the theoretical diversity of games, although some of the most common games that we see in the literature are conversation-based games (e.g., negotiation, social deduction)

and navigation games. These games often introduce new aspects to agent interactions like: multi-step episodes, multi-agent interactions, non-linguistic actions, and embodiment.

These kinds of systems, as a whole, are somewhat less popular in the literature. On a practical level, more complex systems are more difficult to implement and even harder to get to converge reliably—many higher-level behaviors, such as planning or inferring other agent's knowledge, are difficult problems for reinforcement learning in general, let alone with discrete multi-agent emergent communication. On a methodological level, more complexity in the ECS makes it harder to formally analyze the system as well as eliminate confounding factors in empirical investigation. With so many moving parts, it can be difficult to prove that some observed effect is not just a result of some seemingly innocent hyperparameter choice (e.g., learning rate, samples in the rollout buffer) [Boldt and Mortensen, 2023]. Nevertheless, we have reason to believe that these complexities are critical to understanding and learning human language as a whole [Bisk et al., 2020], meaning that the difficulties of more complex systems are worth overcoming as they are part of the process of creating more human-like emergent languages, which are more informative for learning about human language and more suitable for applications in NLP.

Grid-world navigation "Generalizing Emergent Communication" [Unger and Bruni, 2020, BSD-3-clause license] introduces an ECS which takes some of the basic structure of the signalling game and applies it to a navigation-based system derived from the synthetic Minigrid/BabyAI environment [Chevalier-Boisvert et al., 2018, 2023]. A sender with a bird's-eye view of the environment sends messages to a receiver with a limited view who has to navigate to a goal location. Beyond navigation, some environments present a locked door for which the receiver must first pick up a key in order to open. What distinguishes this system most from the signalling game is that it is multi-step and embodied such that the utterances within an episodes are dependent on each other. Among other things, this changes the distribution properties of the utterances. For example, if the receiver is in Room A at timestep T, it is more likely to be in Room A at timestep $T + 1$; thus if utterances are describing what room the receiver is in, this means that an utterance at $T + 1$ has less uncertainty given the content of an utterance at T. Practically speaking, the multiple utterances in a given episode are concatenated together to form a single line in the corpus in order to maintain the dependence of later utterances on previous ones.

Continuous navigation "Mathematically Modeling the Lexicon Entropy of Emergent Language" [Boldt and Mortensen, 2023, GPL-3.0 license] introduces a simple navigation-based ECS which is situated in a continuous environment. A "blind" receiver is randomly initialized in an obstacle-free environment and must navigate toward a goal zone guided by messages from the sender which observes the position of the receiver relative to the goal. The sender sends a single discrete token at each timestep, and a line in the dataset consists of the utterances from each timestep concatenated together. This system shares the time-dependence between utterances of the grid-world navigation system although with no additional complexity of navigating around obstacle, opening doors, etc. On the other hand, the continuous nature of this environment provides built-in stochasticity since there

	min	25%	50%	75%	max
Token Count	48616	67248	110000	1061520	42977805
Line Count	999	5765	10000	10000	2865187
Tokens per Line	5.87	7.00	11.00	33.53	7212.72
Tokens per Line SD	0.00	0.00	2.31	13.81	445.84
Unique Tokens	2	7	10	20	902
Unique Lines	18	1253	2440	4911	309405
1-gram Entropy	0.36	2.12	2.80	3.37	6.60
1-gram Normalized Entropy	0.16	0.71	0.82	0.90	1.00
2-gram Entropy	0.42	3.16	4.11	5.88	12.88
2-gram Conditional Entropy	0.06	0.85	1.41	2.54	6.29
Entropy per Line	4.38	21.23	30.80	71.85	30233.52

Table 2.2: Five-number summary of the analyses across corpora of ELCC. Entropy in bits.

are (theoretically) infinitely many distinct arrangements of the environment that are possible, allowing for more natural variability in the resulting language.

Social deduction "RLupus: Cooperation through the emergent communication in The Werewolf social deduction game" [Brandizzi et al., 2022, GPL-3.0 license] introduces an ECS based on the social deduction game *Werewolf* (a.k.a., *Mafia*) where, through successive rounds of voting and discussion, the "werewolves" try to eliminate the "villagers" before the villagers figure out who the werewolves are. In a given round, the discussion takes the form of all agents broadcasting a message to all other agents after which a vote is taken on whom to eliminate. As there are multiple rounds in a given game, this system introduces multi-step as well as multi-speaker dynamics into the language. Furthermore, the messages also influence distinct actions in the system (i.e., voting). These additional features in the system add the potential for communication strategies that are shaped by a variety of heterogeneous factors rather than simply the distribution of observations (as in the signalling game).

2.5 Analysis

In this section we give present a brief set of analyses that demonstrate some of the possible insights that can be gained from ELCC. Table 2.2 shows the five-number summary of the corpus-level metrics in ELCC. The corpora come in all shapes and sizes, so to speak, demonstrating a wide range of token counts, vocabulary sizes, entropies, and so on. The variety, in large part, comes from the diversity of systems included in ELCC rather than variation within a system. Thus research focusing on a single or narrow range of emergent communication systems—the norm prior to ELCC—restricts itself to a limited diversity of corpus "shapes"; ELCC, in turn, provides an easy opportunity to expand the breadth of many such approaches.

The range of analyses ELCC enables is greatly multiplied by a resource like XferBench [Boldt and Mortensen, 2024b], a deep transfer learning-based evaluation metric for emergent

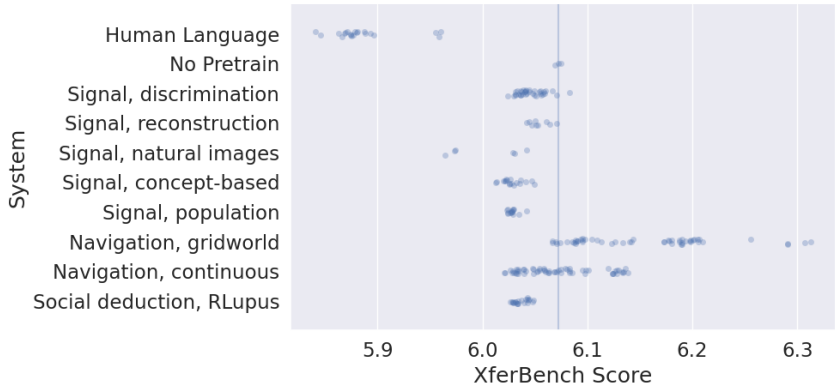


Figure 2.2: XferBench score across ELCC and human language baselines; lower is better. "No pretrain" baseline illustrated with the line on the plot.

```
[47, 2466, 47, 3923, 3325, 3107, 3350, 3923,
    1216, 3980, 1617, 3350, 1897, 556, 0]
[3925, 3925, 3925, 3325, 1172, 2530, 3925, 1209,
    3493, 665, 512, 3923, 2432, 309, 0]
[2128, 2128, 2371, 3925, 946, 512, 1962, 1288,
    2250, 1722, 1722, 1962, 3755, 2695, 0]
```

(a) Best-performing: signalling game [Yao et al., 2022] with the COCO dataset.

```
[3, 3, 3, 3, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7]
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[3, 3, 3, 3, 3, 3, 3, 3]
```

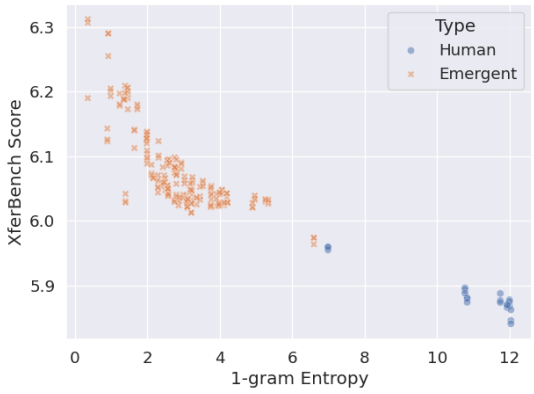
(b) Worst-performing: BabyAI-based navigation game [Unger and Bruni, 2020] (hyperparameters in text).

Figure 2.3: Sample utterances from the best and worst performing emergent language corpora on XferBench from ELCC.

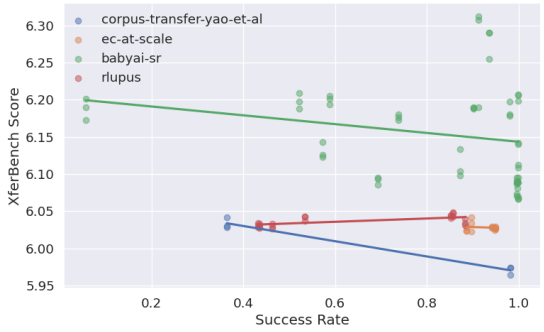
languages. This metric quantifies how good a corpus is as pretraining data for a human language-based downstream task, specifically language modelling (thus a lower score is better). XferBench proves to be particularly powerful for analyzing ELCC because it works in an environment-agnostic way, taking only a corpus of tokenized utterances as input. In fact, ELCC and XferBench permit the first large-scale comparison of emergent language systems with an *evaluative* metric.

Explaining XferBench's performance In addition to the purely descriptive metrics discussed above, we also present evaluative metrics via XferBench in Figure 2.2. We run XferBench three times for each corpus since there inherent stochasticity in XferBench. We see that most of the emergent languages occupy a band which slightly outperforms the baselines (i.e., no pretraining at all) while significantly underperforming human languages (exception discussed below). Notably, two of the environments with the worst-performing corpora are the grid-world [Unger and Bruni, 2020] and continuous [Boldt and Mortensen, 2023] navigation environments, while the signalling games perform better consistently.

Inspecting some utterances from the best- and worst- performing corpora, we can see a qualitative difference in Figure 2.3. The best-performing corpus uses a variety of tokens derived from a large vocabulary (given the high token IDs), while the worst-performing corpus repeats the same two tokens with little variation (this sample is representative of the whole corpus). We hypothesize that pretraining on repetitive strings of a small variety of



(a) Plot of XferBench score versus unigram entropy for emergent languages and baseline human languages from XferBench.



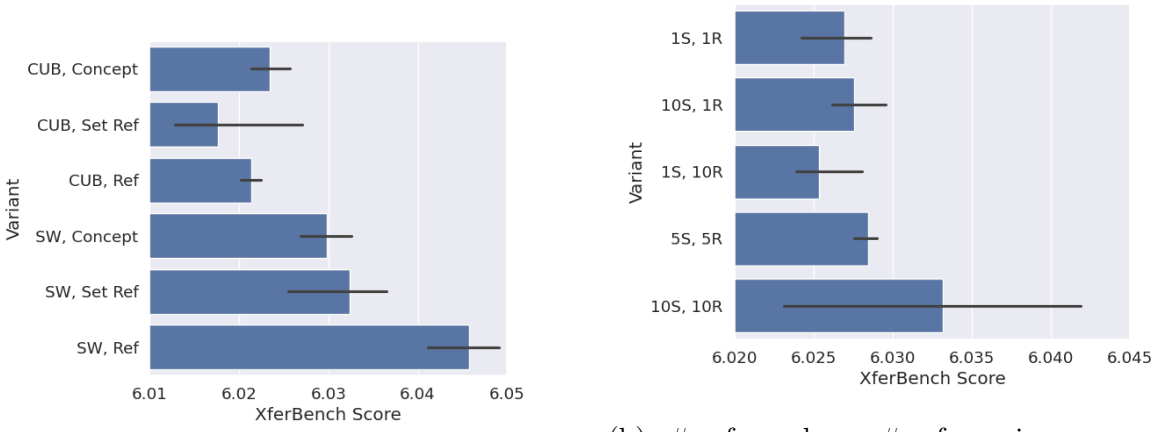
(b) Plot of XferBench score versus success rate, separated by emergent communication system.

Figure 2.4

tokens poorly conditions the model used in XferBench, supported by the fact that the lowest entropy corpora perform the worst on XferBench.

The qualitative analysis suggests that something along the lines of variation or information content might be correlated with XferBench score. To investigate this, we plot two possible explanatory variables against XferBench scores: unigram entropy and task success rate Figure 2.4. Immediately, we can see that there is a strong correlation between entropy and XferBench score. In fact, this plot gives some insight into the anomalously low score on "Signal, natural images" [Yao et al., 2022] and anomalously high score for Hindi (an unresolved quandary of the XferBench paper): both of these corpora perform as expected given their entropies. On the other hand, success rate does not seem to be well-correlated with score on XferBench; surprisingly enough, the worst-performing corpus shown above still sported a >90% task success rate!

Evaluating improvements in ECS design Finally, we are also able to use XferBench and ELCC to evaluate some of the innovations in emergent communication system design made by papers contributing to ELCC. Namely, we look at Mu and Goodman [2021b] and "Emergent Communication at Scale" [Chaabouni et al., 2022]. Mu and Goodman [2021b] introduce (as discussed in Section 2.4) a more sophisticated, concept-focused version of the signalling game, comparing it against a vanilla signalling game ("reference") and an intermediate form of the concept version ("set reference"), finding that the introduced games promote more systematic and interpretable emergent languages. On the other hand, Chaabouni et al. [2022] introduces multi-agent populations to the signalling game but does not find that larger populations have a beneficial effect on communication. Looking at the systems' performance XferBench (Figure 2.5), we can see that the proposed improvements to the signalling game do not have an appreciable effect on XferBench performance in either case. These results do not detract from the original findings; instead, evaluating the design changes with XferBench better contextualizes work, highlighting to what degree certain desirable features of emergent



(a) Expected order: concept, set reference, reference [Mu and Goodman, 2021b].

(b) # of senders, # of receivers; more agents expected to perform better than fewer [Chaabouni et al., 2022].

Figure 2.5: XferBench scores compared to expected order; lower is better.

languages (e.g., interpretability, robustness) correspond with suitability for deep transfer learning.

2.6 Discussion

Work enabled by ELCC In the typical emergent communication paper, only a small amount of time and page count is allocated to analysis with the lion's share being taken up by designing the ECS, implementing it, and running experiments. Even if one reuses an existing implementation, a significant portion of work still goes towards designing and running the experiments, and the analysis is still limited to that single system. While this kind of research is valid and important, it should not be the only paradigm possible within emergent communication research. To this end, ELCC enables research which focus primarily on developing more in-depth analyses across a diverse collection of systems. Furthermore, removing the necessity of implementing and/or running experiments allows researchers without machine learning backgrounds to contribute to emergent communication research from more linguistic angles that otherwise would not be possible.

In particular, ELCC enables work that focuses on the lexical properties of emergent communication, looking at the statical properties and patterns of the surface forms of a given language (e.g., Zipf's law [Zipf, 1949]). Ueda et al. [2023] is a prime example of this; this paper investigates whether or not emergent languages obey Harris' Articulation Schema (HAS) by relating conditional entropy to the presence of word boundaries [Harris, 1955, Tanaka-Ishii, 2021]. The paper finds mixed evidence for HAS in emergent languages but only evaluated a handful of settings in a single ECS, yet it could be the case that only systems with certain features generate languages described by HAS. The variety of systems provided by ELCC could, then, provide more definitive empirical evidence in support or against the presence of HAS in emergent languages. Additionally, ELCC can similarly extend the range of emergent languages evaluated in the context of machine learning, such as Yao et al. [2022],

Boldt and Mortensen [2024b] which look at emergent language's suitability for deep transfer learning to downstream NLP tasks or van der Wal et al. [2020] which analyzes emergent languages with unsupervised grammar induction.

ECS implementations and reproducibility In the process of compiling ELCC, we observed a handful of trends in the implementations of emergent communication systems. A significant proportion of papers do not publish the implementations of experiments, severely limiting the ease of reproducing the results or including such work in a project such as ELCC, considering that a large amount of the work in creating an ECS is not in the design but in the details of implementation. Even when a free and open source implementation is available, many projects suffer from underspecified Python dependencies (i.e., no indication of versions) which can be difficult to reproduce if the project is older than a few years. Furthermore, some projects also fail to specify the particular hyperparameter settings or commands to run the experiments presented in the paper; while these can often be recovered with some investigation, this and the above issue prove to be obstacles which could easily be avoided. For an exemplar of a well-documented, easy-to-run implementation of an ECS and its experiments, see Mu and Goodman [2021b] at `https://github.com/jayelm/emergent-generalization/` which not only provides dependencies with version and documentation how to download the data but also a complete shell script which executes the commands to reproduce the experiments.

Future of ELCC While ELCC is a complete resource as presented in this paper, ELCC is intended to be an ongoing project which incorporates further ECSs, analyses, and taxonomic features as the body of emergent communication literature and free and open source implementations continues to grow. This approach involves the community not only publishing well-documented implementation of their ECSs but also directly contributing to ELCC in the spirit of scientific collaboration and free and open source software. ELCC, then, is intended to become a hub for a variety of stakeholders in the emergent communication research community, namely a place for: ECS developers to contribute and publicize their work, EC researchers to stay up-to-date on new ECSs, and EC-adjacent researchers to find emergent languages which they can analyze or otherwise use for their own research.

Limitations Emergent communication research is primarily basic research on machine generated data; thus, ELCC has few, if any, direct societal impacts. From a research point of view: while ELCC attempts to provide a representative sample of the ECSs present in the literature, it is not comprehensive collection of all of the open source implementations let alone all ECSs in the literature. This limitation is especially salient in the case of foundational works in EC which have no open source implementations (e.g., Mordatch and Abbeel [2018]). Thus, the contents of ELCC could potentially result in an over-reliance on the particular systems included resulting in an unfamiliarity with the data and limiting research on those currently not included in ELCC. Including the data-generating code and metadata describing the systems in ELCC has partially addressed this issue, and future work adding more open source implementations and reimplementing seminal papers could continue to ameliorate this limitation.

Beyond the variety of systems, in its design ELCC only provides unannotated corpora without any reference to the semantics of the communication, which limits the range of analyses that can be performed. For example, measures of compositionality, such as topographic similarity [Brighton and Kirby, 2006a, Lazaridou et al., 2018b], are precluded because they fundamentally a relationship between surface forms and their semantics. In terms of compute resources, we estimate that on the order of 150 GPU-hours (NVIDIA A6000 or equivalent) on an institutional cluster were used in the development of ELCC, and additional 1000 GPU-hours were used to generate the results of XferBench on ELCC. This research could be difficult to reproduce without access to institutional resources.

2.7 Conclusion

In this paper, we have introduced ELCC, a collection of emergent language corpora annotated with taxonomic metadata and suite of descriptive metrics derived from free and open source implementations of emergent communication systems introduced in the literature. ELCC also provides code for running these implementations, in turn, making those implementations more reproducible. This collection is the first of its kind in providing easy access to a variety of emergent language corpora. Thus, it enables new kinds of research on emergent communication which involve a wide range of emergent communication, focusing directly on the analysis of the emergent languages themselves.

Chapter 3

Adding Semantic Annotations to Corpora [proposed]

3.1 Introduction

ELCC enables statistical analyses to be performed across an unprecedented variety of emergent language corpora with ease. Nevertheless, the fact that corpora of ELCC only contain utterances with no accompanying context severely limits the range of analyses that can performed in the grand scheme of emergent communication research. The surrounding context of the utterance is necessary for investigating the semantics of utterances which in turn are necessary for understanding the syntax, pragmatics, and broader social context of the utterance—all major areas of interest for emergent communication research. Thus, we propose ELCC Plus which expands upon ELCC primarily by adding context to each utterance in the corpora, capturing information such as the state of the world, identity of the speaker, the speaker's observation, the previous and following timesteps, and the progress in the overall optimization process. Providing this additional context in an easily-accessible format based on ELCC will enable a wide range of analyses from the emergent communication literature to be performed directly on the static data without needing to spin up an emergent communication system directly.

Related work The most closely related work comes from the standardized reinforcement learning toolkits PettingZoo [Terry et al., 2021] and Minari [Younis et al., 2024].[1] Terry et al. [2021] introduce the Agent–Environment Cycle (AEC) formalism to describe an API for multiagent reinforcement learning environments with a significant amount of generality. AEC, and consequently the API of PettingZoo, should, in theory, be able to express emergent communication environments. Leveraging this formalism and API would improve interoperability with existing tools in the RL ecosystem, but it could also come at a cost of naturally expressing emergent communication environments. Minari, on the other hand, is a toolkit for offline RL, including the serialization of trajectories (a.k.a., episodes, rollouts) into static datasets. Minari currently does not have any support for PettingZoo/AEC or any

[1] Both of these are projects of the Farama Foundation which also maintains Gymnasium, the somewhat official continuation of OpenAI Gym.

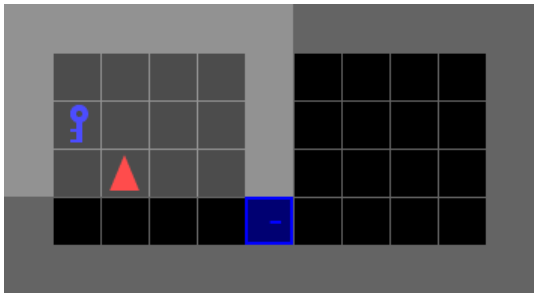


Figure 3.1: A visualization of the Minigrid (f.k.a., BabyAI) navigation environment [Chevalier-Boisvert et al., 2018, 2023].

other multi-agent environments. Nevertheless, its serialization techniques will be relevant even if the codebase is not used directly.

3.2 Format

When designing the format for the rich corpora, there are two primary goals in representing the episodes in the emergent communication systems. The first goal is to completely and accurately represent what happens in the episode. The second is to do so in a consistent format with consistent interpretations of the elements of the format. Due to the diverse ways in which ECSs can vary, this is a difficult, if not impossible, goals to achieve with a simple, static serialization. Completely representing each ECSs would result in largely inconsistent formatting while adhering to stricter formatting would result in a lossy storage of the ECS episodes.

Thus, we propose a two-part format for representing the corpora of ELCC Plus. The first part is a serialization of the episodes/trajectories unique to each ECS, while the second part is a collection of (Python) functions which provide a consistent interface to various parts of the corpus. In particular, the serialization will capture as much information as possible about each episode, adhering to conventions (e.g., utterances are always referred to as "utterance", not "message" while not being limited to a particular format (e.g., each timestep could have one, zero, or many utterances). While the serializations will be as consistent as possible, the level of variation required to accurately represent an ECS episode will require programmatic adapters to provide a consistent interface for the elements the ECSs do have in common. This approach yields the best of both worlds: the ECS-specific serialization format allows as much information to be represented as possible in a way that corresponds well to the environment while the adapter-based interface allows for easy extraction of relevant data across the variety of corpora. Furthermore, this format is extensible since new adapters can be written to extend the interface without needing to rerun the underlying ECS.

Concrete Example

Environments In order to illustrate the format of the enriched collection, we will use two environments. The first environment is the discrimination variant the signalling game where the observations are concatenations of one-hot vectors. The second environment is

a sender–receiver navigation based on Minigrid framework. In this game, the receiver is an embodied agent in a grid world environment who must navigate to the goal based on messages from the sender who can observe the whole environment (illustrated in Figure 3.1). Additionally, there is a one-to-many relationship between messages and actions the receiver takes, that is, the receiver may take multiple steps before the sender sends another message.

Serialization In Figure 3.2, we present a simplified serialization of episodes in the signalling game and navigation environments. For the signalling game, the serialization is relatively straightforward since every episode consists of a single timestep with the same components; there are observations and actions made by the sender and the receiver, a reward given by the environment, and other metadata like the step in the optimization process. The navigation environment has more moving parts, and the serialization is, as a result, more complicated. The observations themselves have more information and structure compared to the single vector in the signalling game. More important to studying communication, though, is the fact that there are multiple time steps in the environment which themselves potentially contain utterances.

Even in this simple example, it is evident that coming up with a strict data format for serializing emergent communications systems would difficult and cumbersome. The signalling game only consists trivially of one or two steps, and it is much more natural to represent as not having any steps at all. Meanwhile the navigation environment requires being broken into a variable number of steps. If we were to try to serialize a multi-agent navigation environment, we would no longer able to simply refer to the `utterance` or `receiver_action` of each step and instead would have to generalize the representation to multiple utterances and actions per step. This list of generalization can go on indefinitely, and so it is the best interest of the collection to keep this part of the representation flexible so as to not limit the future scope of the collection.

Interface The interface we will introduce for illustration purposes will consist of three elements which provide: (1) the utterances alone, (2) utterances with meaning the sender is trying to convey, and (3) utterances with the meaning as interpreted by the receiver. While these elements are simple, the demonstrate nicely how the implementations adapt the variability in the serialization the uniformity required by the interface. We give Python pseudo-code for the implementation of this interface in Figure 3.3.

The interface implementation for the signalling game is, again, relatively straightforward due to the simplicity of the environment and, in turn, serialization. On the other hand, the implementations for the navigation environment have to account for more variability and nuance. Firstly, the implementations for the utterances as well as utterances plus sender meaning requires that steps without utterances be filtered out. More interestingly, the receiver-side semantics implementation requires aggregating the actions the receiver takes over multiple timesteps as the interpretation of the sender's utterance may entail taking a sequence of actions.

One could easily quibble here with the interpretation of "meaning" in the interface's implementations, but this illustrates an important benefit of the two-part representation proposed. So long as the initial serialization format is expressive enough for a given emergent

```yaml
- optimization_step: 100
  reward: 1.0
  utterance: [3, 1, 4, 1]
  observation:
    sender: [1, 0, 1, 0]
    receiver:
      - [1, 0, 0, 1]
      - [1, 0, 1, 0]
      - [0, 1, 0, 1]
  correct_idx: 1
  receiver_action: 1

- optimization_step: 101
  reward: 0.0
  utterance: [1, 2, 3, 4]
  observation:
    sender: [1, 0, 0, 1]
    receiver: ...
  correct_idx: 1
  receiver_action: 0
```

(a) Basic signalling game

```yaml
- optimization_step: 100
  reward: 0.0
  steps:
    - map:
        walls: ...
        key: ...
        receiver: ...
        door: ...
        goal: ...
      utterance: [1, 1, 2, 2]
      receiver_action: move
      observation:
        sender: ...
        receiver: ...
      stop: false

    - map: ...
      utterance: null
      receiver_action: pick_up
      observation: ...
      stop: false

    - map: ...
      stop: true

- optimization_step: 101
  reward: 1.0
  steps: ...
```

(b) Multi-step navigation game based on Minigrid environment

Figure 3.2: Example representations of two different emergent communication environments. Format is based on YAML.

communication system, the more nuanced discussions (e.g., what constitutes a meaning in this environment?) can take place at the level of the interface's implementation (or initial definition). Importantly, this does not require rerunning the emergent communication system and recollecting the data; rather, the implementation of the interface can be edited as necessary and be rerun over the serialization (which is comparatively cheap computationally to running a deep learning-based multi-agent simulation).

Contents

The contents of the enhanced corpora collection will largely overlap with ELCC, as the implementations for those emergent communication systems have already been verified as runnable. The contents of this collection, though, will be extended, as necessary, to illustrate elements of the interface as well as to provide meaningful data for the experiments illustrating the utility of the proposed collection and its format.

```
def utterances(episodes):
  for ep in episodes:
    yield ep.utterance

def semantics_sender(episodes):
  for ep in episodes:
    yield ep.observation.sender

def semantics_receiver(episodes):
  for ep in episodes:
    idx = ep.receiver_action
    yield ep.observation.receiver[idx]
```

(a) Signalling game

```
def utterances(episodes):
  for ep in episodes:
    for step in ep.steps:
      if step.utterance is not null:
        yield step.utterance

def semantics_sender(episodes):
  for ep in episodes:
    for step in ep.steps:
      if step.utterance is not null:
        yield ep.utterance, ep.observation.sender

def semantics_receiver(episodes):
  for ep in episodes:
    actions = []
    utterance = ep.steps[0].utterance
    for step in ep.steps:
      if step.utterance is not null or step.stop:
        yield utterance, actions
        actions = [step.receiver_action]
      else:
        actions.append(step.receiver_action)
```

(b) Navigation game

Figure 3.3: Python pseudo-code for interfaces corresponding to the serializations in Figure 3.2.

3.3 Experiments

The experiments for this chapter will primarily demonstrate how it is possible to compute a wide range of metrics from the emergent communication literature based on the ELCC Plus. These experiments will highlight that ELCC Plus enable performing a rich set of analyses over a variety of emergent languages based on static data. This breaks the current paradigm of usually needing to run the emergent communication system itself in order to generate the required data for analyses involving the semantics of emergent communication. The following metrics are proposed.

- *Topographical similarity* quantifies compositionality by measuring correlation between pairwise distances in the message space and pairwise distances in the meaning space. Also known as *toposim*. [Brighton and Kirby, 2006a, Lazaridou et al., 2018b]
- *Instantaneous coordination* measures the effect that one agent's actions have on another's actions in a multi-agent environment. It is quantified as the mutual information between the utterances of a first agent and the action of a second agent at the next timestep. [Jaques et al., 2019]
- *Diachronic features* refer to any metric that is tracked over the course of the optimization process. For example, the entropy of communication may start with a high entropy as it is mostly random but gradually decrease as the agents develop a communication protocol with a limited number of expressions.

- *Ease-of-teaching* measures how quickly and effectively an emergent language is acquired by new member in a dynamic population of agents. [Li and Bowling, 2019]

Chapter 4

Evaluation of Emergent Languages with Deep Transfer Learning

Abstract

In this chapter, we introduce XferBench, a benchmark for evaluating the overall quality of emergent languages using data-driven methods. Specifically, we interpret the notion of the "quality" of an emergent language as its similarity to human language within a deep learning framework. We measure this by using the emergent language as pretraining data for a downstream NLP tasks in human language—the better the downstream performance, the better the emergent language. We implement this benchmark as an easy-to-use Python package that only requires a text file of utterances from the emergent language to be evaluated. Finally, we empirically test the benchmark's validity using human, synthetic, and emergent language baselines.[1]

4.1 Introduction

Neural language models learn many things in pretraining, but research suggests [Artetxe et al., 2020] that a substantial part of that knowledge is not simply knowledge of a particular language or domain, but rather knowledge of "how to language." We currently teach models to "language" using vast quantities of text dredged from the dark recesses of the Web—text that is full of bias, toxicity, and potential intellectual property violations. Ideally, we would be able to teach models to "language" without such compromises through the use of synthetic data, but mainstream approaches to synthesizing data produce outputs that do not have the same structural and social properties as human language.

Emergent communication (EC), also called emergent language (EL), is a potential solution to this problem [Yao et al., 2022, Downey et al., 2023, Mu et al., 2023]. Emergent languages are communication systems developed *de novo* among multiple agents in a reinforcement learning simulation. Because the conditions under which they develop mirror, reductively, the conditions under which languages develop among humans, there is reason to believe that ELs

[1]Based on "XferBench: a Data-Driven Benchmark for Emergent Language" appearing in the *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* [Boldt and Mortensen, 2024b].

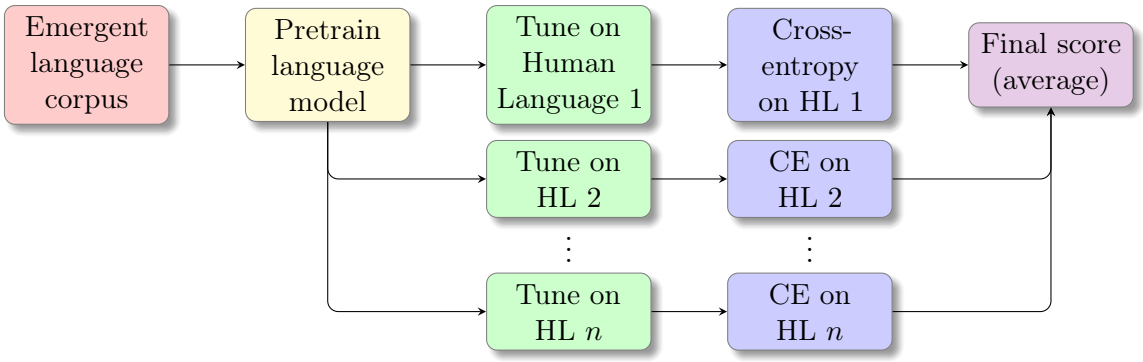


Figure 4.1: Illustration of the architecture of XferBench.

will ultimately be more like human language than other sources of synthetic data. However, up to this point, there is no way of quantifying—in a holistic way—how much like human languages any particular EL really is, or to what extent it may provide useful pretraining signals.

Research on deep learning-based emergent communication has seen the introduction of many metrics to measure various aspects of the language. These metrics quantify notions such as compositionality [Brighton and Kirby, 2006a, Lazaridou et al., 2018b], expressivity [Guo et al., 2023], ease-of-teaching [Li and Bowling, 2019], and zero-shot transfer [Bullard et al., 2020], to name a few. Despite this proliferation of metrics, emergent language largely lacks *evaluation* metrics. An evaluation metric is specifically one that measures the *overall quality of an emergent language* and not simply a particular property. Thus, we introduce XferBench, a data-driven benchmark for evaluating the overall quality of emergent languages using transfer learning with deep neural models.

Evaluation metrics are critical in gauging progress in technical fields since they quantify otherwise vague notions of improvement over time. Benchmarks, in particular, pair evaluation metrics with specific data and evaluation procedures to compare various systems on common ground. Benchmarks and shared tasks have been critical to the development of NLP from the Penn Treebank [Marcus et al., 1993] to the WMT datasets [Bojar et al., 2014] to GLUE [Wang et al., 2018].

In the field of emergent communication specifically, Yao et al. [2022] introduced the idea of using *corpus transfer* as means of practically applying emergent communication to deep learning-based NLP via transfer learning. In corpus transfer, a language model is pretrained on a corpus of emergent language utterances before being tuned on real data for a human language-based downstream task. As a corollary, they suggest that the effectiveness of this transfer can serve as a means of evaluating the quality of the emergent in a more general sense. This is based on the intuition that the more similar two language are, the better transfer learning works from one to the other (observed in Zoph et al. [2016], for example).

This chapter takes the transfer learning-as-an-evaluation metric idea from Yao et al. [2022] and expands it into a full benchmark, XferBench, for emergent languages (illustrated in Figure 4.1). An evaluation metric for emergent languages in a benchmark format is the first of its kind. Additionally, XferBench is unique within emergent communication for being primarily data-driven instead of relying on particular handcrafted algorithms for quantifying a given phenomenon. This means that XferBench can be easily scaled up in the future as

the field of emergent communication advances and requires expanded means of evaluating emergent languages. Finally, XferBench is distributed as a user-friendly Python package, allowing researchers from across the field of emergent communication to apply XferBench to their own work on emergent communication.

Contributions This chapter makes the following contributions: (1) Introduces XferBench, a data-driven benchmark for evaluating the overall quality of an emergent language, the first of its kind in emergent communication. (2) Provides a analysis of the quality human, synthetic, and emergent language according to XferBench. (3) Provides an easy-to-use Python implementation of XferBench.

4.2 Related Work

Emergent Communication This chapter is situated in the field of emergent communication (a.k.a. emergent language) which is generally covered by the review Lazaridou and Baroni [2020]. The field centers around the invention of language by deep neural networks typically using multi-agent reinforcement learning techniques. The study of emergent communication is intended to (1) shed light on the origin and nature of the human language [LaCroix, 2019, Moulin-Frier and Oudeyer, 2020, Galke et al., 2022] and (2) provide an alternative approach to problems in NLP and multi-agent reinforcement learning which relies on constructing language from the ground up and not just pre-existing (human) languages alone [Li et al., 2020, Yao et al., 2022, Mu et al., 2023, Downey et al., 2023].

Transfer Learning Transfer learning for deep neural networks is a key component of XferBench and follows in general tradition of Zoph et al. [2016]. Specifically, this chapter draws heavily from Yao et al. [2022] (see also Papadimitriou and Jurafsky [2020], Artetxe et al. [2020]) which introduce the technique of *corpus transfer* for emergent language, that is, pretraining a neural model on an emergent language corpus before tuning it on a downstream human language task. In particular, this chapter takes Yao et al. [2022]'s idea of using corpus transfer as a metric and adapts it into a benchmark pipeline which can easily be applied to new emergent languages.

Benchmarks Work such as Guo et al. [2023] and Perkins [2022] have looked at benchmarking particular aspects of emergent languages, but XferBench is the first of its kind in benchmarking the overall quality of an emergent language. Yao et al. [2022] also explicitly provide a metric for emergent language quality, but this metric is restrictive in that it can only be applied to emergent languages derived from a model that takes images (that have captions available) as input; this conflicts with the design goals of XferBench discussed below.

Outside of emergent communication, XferBench is more analogous to benchmarks for generative models (e.g., Fréchet Inception Distance [Heusel et al., 2017] for image generation) than more traditional NLP benchmarks like GLUE [Wang et al., 2018] or SQuAD [Rajpurkar et al., 2016]. This is because emergent communication is a generative enterprise, where one of the main goals is to create samples (emergent languages) which resemble a target distribution (human languages) either generally or in some particular respect. Furthermore,

metrics like FID are primarily self-supervised, data-driven measures of similarity in the same vein as XferBench. This is in contrast to more traditional NLP benchmarks which combine data-driven methods with many human judgments (i.e., through labeled examples).

4.3 XferBench

Design Goals

We frame the primary design goals of the benchmark as three desiderata:

- **D1** Quantitatively capture a meaningful notion of the overall quality[2] of an emergent language from a data-driven perspective.
- **D2** Be applicable to as wide a variety of emergent languages as possible, not restricted to a specific game, environment, or agent architecture.
- **D3** Be relevant and accessible to the broader EC/EL community, by being: (a) easy to interpret, (b) minimally biased with regards to language typology, (c) runnable with minimal coding experience, and (d) runnable on modest hardware.

While there are other consideration in the benchmark, these form the bulk of the motivation. In the following paragraphs we expand upon the motivation for each design goal.

D1: Quantifying quality D1 is the core of what a benchmark seeks to do: to quantify a desirable property of a given system such that it can be compared directly to other systems (i.e., be an *evaluation* metric). There are two distinct senses in which XferBench strives towards this goal. First, XferBench measures how good an emergent language is from a specifically machine learning perspective; that is, it addresses the question, "How useful would this emergent language be for practical machine learning tasks?" The second sense is more general: XferBench addresses the question, "How similar is an emergent language to human language according to how deep neural networks process language?" That is, it uses data-driven techniques to quantify the similarity between emergent language and human language in some general sense.

D2: Wide applicability D2 is intended to make XferBench practically applicable to a wide range of EC research. The field of EC has an especially diverse set of possible approaches, environments, agents, games, etc. Thus, it is especially salient that the benchmark be designed with interoperability in mind, having minimal assumptions as to the nature of the EC system being evaluated.

The influence of this design goal is primarily seen through the use of a textual corpus as the sole input to the benchmark: the vast majority of EC systems generate utterances which can be represented as sequences of discrete tokens.[3] EC presents the opportunity for much richer representations of its language: leveraging the grounded semantics of the communication, incorporating non-verbal behavior, and even directly interacting with the

[2]We are aiming for *a* meaningful notion of overall quality: we are not claiming that this is the only meaningful notion nor that it is the best among all possible notions of "quality".

[3]In the minority case, there are EC methods which use communication channels that are, for example, continuous [Eloff et al., 2021] or even pictorial [Mihai and Hare, 2021b].

agents themselves. Yet such richer representations also limit the range of EC systems to which XferBench could apply. Even if it is possible to define some universal EC interface that could allow for richer representations, the implementation cost for each and every EC system to be tested is significant compared to the ease of producing a corpus of utterances from the emergent language.

D3: Easy-to-use D3 is critical to the success of XferBench as a practical tool for diverse field of researchers—a benchmark is expressly *for* the broader research community, and, as such, should be widely accessible. In particular, D3a demands that XferBench be conceptually simple with results that can easily be reported, compared, and incorporated into a research program. D3b is relevant to both aspects of D1. First, if XferBench is to gauge an EL's practical use in machine learning, it should seek to use a typologically diverse set of human languages in the downstream tasks. Second, since XferBench is trying to capture a notion of "similarity to human language generally", it is important to test this against a wide range of language typologies so as not to unnecessarily narrow the criteria for "similar to human language". D3c is particularly important for incorporating interdisciplinary researchers into the field of EC who might not have a background in computer programming. Finally, D3d ensures that XferBench is accessible not only to labs and researchers with fewer financial resources but also makes it much easier to incorporate into the fast-paced research and development cycles prevalent in contemporary ML reserach.

Methods

The following procedure describes the benchmark (illustrated in Figure 4.1):

1. Initialize a causal language model.
2. Train the model on the corpus of utterances from the EL being evaluated.
3. Re-initialize the input and output (i.e., language modelling head) embedding layers; this is the *base model*.
4. For each downstream human language:
    a) Train the base model on the human language data.
    b) Evaluate the cross-entropy on a held-out test set of the human language.
5. Average the cross-entropies across the downstream human languages; this is the corpus's score on the benchmark (lower is better).

The structure of the benchmark is derived from the *corpus transfer* method presented in Yao et al. [2022].

Task For XferBench's evaluation task, we choose causal language modeling for a few different reasons. In principle, language modeling is a component of a wide variety of NLP tasks, especially generative tasks; the prevalence of language modeling is in line with the benchmark providing a very general notion of quality that will be familiar to anyone acquainted with NLP. On a practical level, language modeling is easy to acquire data for—especially helpful for evaluating against low-resource languages—and there are fewer hyperparameters and confounding variables compared to other downstream tasks like machine

translation or question-answering. The main limitation from using language modeling is that it itself is not a widespread downstream task and so cannot guarantee direct correlation with metrics on more concrete downstream tasks (e.g., accuracy on a QA task).

For the pretraining task we also use causal language modeling. Due to requiring a wide applicability across emergent languages (Design Goal 2), we select causal language modeling for our pretraining task since it requires only a corpus without any additional annotations or stipulations.

Data The data for the transfer learning targets (viz. human languages) comes from Wikipedia dumps [Wikimedia Foundation] (under the GFDL and CC-BY-SA 3.0 License) hosted by Hugging Face[4]. This dataset provides a diverse set of languages each with sufficient amounts of data. For our downstream human languages, we use the same 10 languages presented in Yao et al. [2022], namely: Basque, Danish, Finnish, Hebrew, Indonesian, Japanese, Kazakh, Persian, Romanian, and Urdu. Having a variety of languages reduces the likelihood that XferBench will be biased toward specific typologies of human language (Design Goal 3b).

We use 15 and 2 million tokens for the pretraining and fine tuning phases, respectively following Yao et al. [2022]. Datasets are always repeated or truncated to fit the required size so that the number of training steps stays constant.

Tokenization For tokenization we use byte pair encoding (BPE) [Gage, 1994a] with a vocabulary size of 30 000 for all human languages. Using BPE across all human languages is done primarily to simplify the implementation and keep tokenization methods consistent across all of the selected human languages. Emergent languages are generally considered to be pre-tokenized since most communication channels consist of one-hot vectors; thus, no additional tokenization or preprocessing is applied.[5]

Model For our model, we use a small configuration of GPT-2 [Radford et al., 2019], similar to that used in Yao et al. [2022]: 6 attention heads, 6 layers, context length of 256, and hidden size of 768 with the remainder of the model parameters being the same as the defaults in the Hugging Face Transformers implementation.[6] This yields 65 million parameters in total. We kept the model on the smaller size to better suit it for the generally small amounts of data emergent languages corpora provide as well as to be more accessible (Design Goal 3d). Further details are listed in Section B.1.

Metric Given the use of language modeling for our evaluation task, we use token-level cross-entropy as the evaluation metric on the downstream task. This is a very common metric, making the outputs easy to interpret (Design Goal 3a). Although perplexity is more common

[4] https://huggingface.co/datasets/wikimedia/wikipedia/tree/97323c5edeffcf4bd6786b4ed0788c84abd24b03

[5] Whether the tokens of an EL should be treated as words or subword units is an open question, although tokens as words is more common (but see Ueda et al. [2023] for tokens as subword units). Practically speaking, many emergent languages are small enough that applying a 30 000-item BPE model would severely reduce the corpus size.

[6] https://huggingface.co/docs/transformers/v4.36.1/en/model_doc/gpt2#transformers.GPT2Config

as an evaluation of language models, the exponential nature of perplexity leads to more circuitous analyses and interpretation in our case, whereas cross-entropy is comparatively linear and additive (loosely speaking).[7] For the final score of the benchmark, we take the arithmetic mean of the cross-entropy across the 10 downstream human languages. That is, we define the benchmark's score for a given source language s as as h_s:

$$h_s = \operatorname*{mean}_{t \in T} (h_{s,t}) \tag{4.1}$$

where $h_{s,t}$ is the test cross-entropy of a model trained on source language s and finetuned and tested on target language t; T is the set of target languages. Since the score is based on cross-entropy, a lower score means better performance.

Implementation

XferBench is implemented as a small Python codebase which relies primarily on Hugging Face Transformers [Wolf et al., 2019] (Apache-2.0 license) and PyTorch [Paszke et al., 2019] (BSD-3-Clause license) libraries. To run the benchmark, all that is required is to install the environment with either pip or conda, and run `python -m xferbench path/to/corpus.jsonl` (Design Goal 3c). The input corpus is simply formatted as a newline-separated list of integer arrays, specifically in the JSON Lines format (see Section B.2 for an example); a Hugging Face dataset (backed by Apache Arrow) can also be used for larger input corpora. The script executes all of the steps of the benchmark and yields a single floating point number which is that corpus's score on XferBench (the benchmark also saves the individual score across target languages for further analysis). Finer-grained functionalities are available and documented in the codebase. The benchmark takes about 5.5 hours to run on a single NVIDIA GeForce RTX 2080 Ti: 90 minutes to train the base model and 30 minutes for tuning and testing on each of the target languages (Design Goal 3d). Since the model is tuned independently on each target language, it is easy to parallelize this step and drastically shorten the wall-clock time of XferBench.

The implementation is available at `https://github.com/brendon-boldt/xferbench` under the MIT license.

4.4 Experiments

Procedures

XferBench The causal language modeling experiment is simply running XferBench as described in Section 4.3 on the reference and emergent languages discussed in Section 4.4.

Machine translation The machine translation experiment is structured similarly to XferBench except with the downstream task being English-to-French translation (using the WMT 2014 dataset [Bojar et al., 2014]). The primary purpose of this experiment is to

[7] For example, it would make more sense to use logarithmic scales and geometric means to average and compare perplexities, but this would just be reverting back to cross-entropy!

determine how well XferBench correlates with a more concrete downstream task (especially one that incorporates language modeling). We choose this language pair in part to gauge the relative differences between the task languages and the baseline human languages (in contrast to XferBench which we want to be largely agnostic to human languages). Looking at our reference human languages, we have: French, the target language itself; Spanish, closely related to French; Russian and Hindi, distantly related to French; and Chinese, Korean and Arabic, not related to French. Instead of using a GPT-2–based model, we use a BART-based model since MT is a conditional generation task (see Section B.1 for details). The pretraining dataset size is increased to 100 million due to the increased difficulty of this task compared to language modeling. We evaluate the translation performance with chrF [Popović, 2015] and BLEU [Papineni et al., 2002] using the default Hugging Face Evaluate metrics (derived from sacreBLEU [Post, 2018]). Evaluation is performed with beam sizes of 1, 3, and 5, and the resulting values are averaged.

We present three settings for this experiment. The first is *Full* which tunes on 50 million source tokens at a higher learning rate ($1 \cdot 10^{-4}$ for training and $2 \cdot 10^{-4}$ for the AdamW optimizer [Kingma and Ba, 2015]), which we found empirically to lead to the best performance. The second is *Frozen*, in which we use the same configuration as *Full* but freeze all but the embedding layers before tuning the model for translation (as in Papadimitriou and Jurafsky [2020], Artetxe et al. [2020]). Finally, we also present *Reduced* which uses a smaller tuning dataset of 10 million tokens and lower learning learning ($2 \cdot 10^{-5}$); the lower rate helped the random baselines converge better as well as showed better distinction between languages.

Reference languages

The following reference languages serve as a way to contextualize the results of XferBench as well as to validate that it is capturing some notion of the quality of the emergent languages (cf. Section 4.4).

Human languages For our baseline line human languages, we selected French, Spanish, Russian, Chinese, Korean, Arabic, and Hindi.[8] Like the evaluation languages, the data is derived from Wikipedia articles (same source as the target languages).

Synthetic languages For synthetic languages, we follow Yao et al. [2022] and use "Zipfian parentheses" from Papadimitriou and Jurafsky [2020]. This synthetic dataset—referred to as *Paren, real*—is hierarchically balanced "parentheses" where each parenthesis is the token ID sampled from the unigram distribution of a human language (hence "Zipfian"). This datasets mimics both the unigram distribution of a human language as well as the basic recursive hierarchical structure. This yields a reasonably strong yet simple baseline for synthetic data.

We also test a fully synthetic dataset (*Paren, synth*) which uses the same hierarchical parenthesis generation script from Papadimitriou and Jurafsky [2020], replacing the data-

[8]The main reason for choosing the high-resource language is due to the higher data requirements of machine translation experiment discussed below.

Setting	Observ.	$\lvert V\rvert$	$\lvert M\rvert$	$\lvert C\rvert$
Disc, small	one-hot	6	11	700
Disc, large	one-hot	100	31	100 M
Recon, large	one-hot	100	31	31 M
Mu+, CUB	embed	20	10	1.3 M
Mu+, SW	embed	14	7	1.2 M
Yao+	embed	4028	15	43 M

Table 4.1: Summary of key hyperparameters in the tested emergent languages. Observations are either one-hot vectors or embeddings. $|V|$, $|M|$, and $|C|$ refer to the vocabulary, message, and corpus size respectively.

derived unigram distribution with Zipf–Mandelbrot distribution:

$$f(w_i) = \frac{1}{(i+\beta)^\alpha} \tag{4.2}$$

where $f(w_i)$ is non-normalized probability weight of word w with 1-based index (rank) i, $\alpha = 1$, $\beta = 2.7$ [Mandelbrot et al., 1953, Piantadosi, 2014].

Random baselines We use two random baselines. The first is simply a uniform unigram distribution across the whole vocabulary with no additional structure (referred to as *Random*). This baseline sheds light on whether the optimization itself, no matter training data, primes the network in some way for transfer learning. The second "random" baseline is no pretraining at all (*No pretrain*); that is, a network which has been freshly initialized at the tuning stage. This baseline helps establish whether or not pretraining on other languages has any impact beyond tuning alone.

Emergent languages

We present a summary of the key hyperparameters of emergent languages in Table 4.1. The emergent language corpora below come from reproductions from existing codebases with the exception of Yao et al. [2022], whose emergent language corpus is available for download. Emergent languages which have a corpus size smaller than the required size are simply repeated and shuffled as many times as necessary so that the model receives the same number of optimization steps.

Generic signalling game The first set of emergent languages we test are generic versions of the of the signalling game (reference game) as implemented in EGG [Kharitonov et al., 2019] (MIT license). These games use one-hot vectors to represent attribute–value observations, that is, observations are elements of the set $V^{|A|}$ where V is the set of values and $|A|$ is the number of attributes. The signalling game is one of the simplest and most used games in emergent communication research.

The first two language are *Disc, small* and *Disc, large* which are two configurations of the discrimination version of the signalling game. Here, the sender makes an observation

and sends a message; then, the receiver must select the corresponding observation from a small set of potential observations (like a multiple-choice question). The *small* configuration consists of 4 attributes and 4 values with a small vocabulary size and medium message length; this setting is intended to represent a toy environment that one might find in an emergent communication paper. The *large* configuration consists of 12 attributes and 8 values with a larger vocabulary and longer message length. Both environments show 5 distractor observations to the receiver (i.e., 6-way multiple choice). Both settings converge to a success rate >95% compared to a random baseline of 17%.

The *Recon, large* environment is based on the reconstruction version of the signalling game. In this version, the receiver does not make any observations and instead must recreate the sender's observation based on the message alone (similar to an autoencoder). The observation space has 8 attributes and 8 values with other settings identical to that of *Disc, large*. Since the reconstruction game considerably harder, the game does not converge but does reach an overall accuracy of 0.014% and per-attribute accuracy of 24% compared to a random baseline of 0.000006% and 13% random baseline, respectively. For details, see Section B.1.

Mu and Goodman [2021b] present the second pair of emergent languages which we test XferBench on (code under MIT license). The emergent communication game is also a discriminative signalling game but with (1) richer observations and (2) more abstract information needing to be communicated. In one setting, the observations are images from ShapeWorld [Kuhnle and Copestake, 2017b] (*Mu+, SW*), a synthetic data of various geometric shapes, and the other setting is CUB [Wah et al., 2011] (*Mu+, CUB*) which contains labeled images of birds; both settings encode features with a CNN which is the passed to the sender and receiver. In the basic discriminative game, the observation made by the sender is the exact same one seen by the receiver. Mu and Goodman [2021b] instead uses a "concept game" where the sender must communicate some abstract concept shared by a set of input images which the receiver will then have to a pick out from a different set of images, some sharing the same concept (e.g., isolating the concept of "triangle" or "bird size"). The ShapeWorld and CUB games had test accuracies of 71% and 66% respectively compared to a random baseline of 50%, comparable to the reported values in the paper. All messages were taken from observations seen in training.

Yao et al. [2022] present a standard discrimination game which uses natural images (Conceptual Captions [Sharma et al., 2018] (images only)) as inputs to the sender and receiver (code unlicensed but distributed on GitHub with paper). The accuracy for the particular emergent language corpus is not reported in the paper, but comparable experiments from the paper would suggest that it converged to an accuracy of >90% compared to a baseline of 0.4% (i.e., 255 distractors).

Hypotheses

The following hypotheses are directly relate to determining whether or not XferBench is quantifying some meaningful notion of the quality of a language (i.e., Design Goal 1).

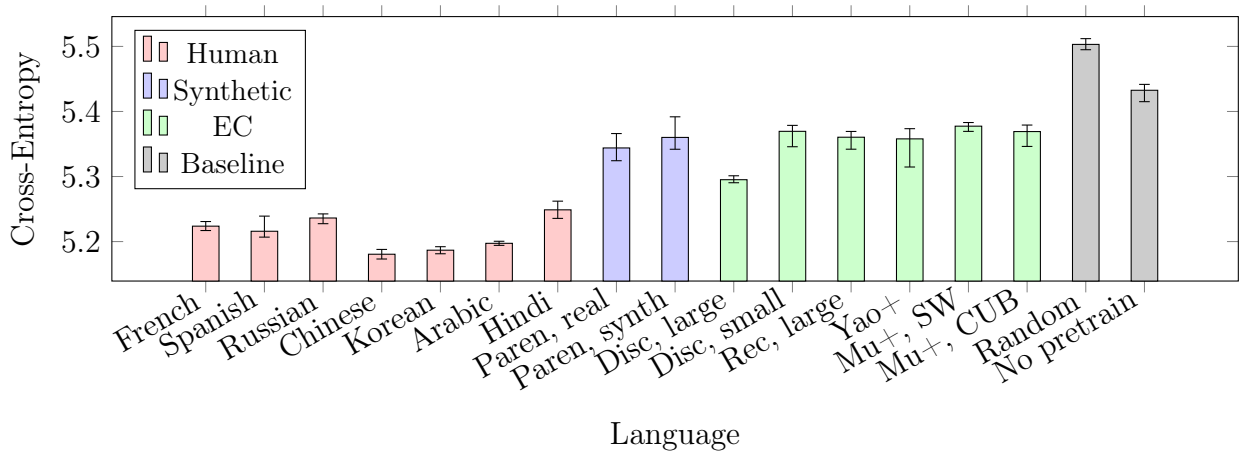


Figure 4.2: Average cross-entropy on target language datasets for each source language. Lower is better. Error bars represent 95% confidence intervals.

(H1) Human languages will perform best, followed by the synthetic and emergent languages, followed by the random baselines.

(H2) Human languages will have similar performance on XferBench (also key for Design Goal 3b); the intuition here is that human languages share deep structural similarities. This hypothesis is supported, in part, by Artetxe et al. [2020]. For the MT experiment, we expect to see the following order of performance based on language relatedness: {*French*}, {*Spanish*}, {*Russian*, *Hindi*}, {*Chinese*, *Korean*, *Arabic*}.

(H3) Languages with a larger vocabulary, longer message length, and larger corpora will perform better. In particular, we expect *Disc, large* will perform better than *Disc, small* since the former is a more "complex" version of the latter. This hypothesis (for vocabulary size and message length) is supported by some experiments in Yao et al. [2022, app. B.4].

(H4) XferBench will correlate well with scores on the machine translation task (i.e., cross-entropy will correlate negatively with chrF).

4.5 Results

XferBench

In Figure 4.2 we show the results of the benchmark (i.e., causal language modeling) on the various baselines. Each mean is displayed with error bars showing the 95% confidence interval of mean as calculated with bootstrapping (details in Section B.5). For reference, the cross-entropies range from about 5.2 to 5.5 corresponding to perplexities of 180 to 240.

The human languages show the best score (lowest cross-entropy) on the benchmark with *Chinese*, *Korean*, and *Arabic* performing the best in one cluster and *French*, *Spanish*, *Russian*, and *Hindi* performing slightly worse in their own cluster (based on confidence intervals). The synthetic and emergent languages all show similar performance with only small variations with the exception of the *Disc, large* language which is better than the rest of the emergent languages but still worse than the human languages. Finally, the random baselines perform

Source	Full	Frozen	Reduced
French	47.8	31.4	35.8
Spanish	48.0	27.9	34.8
Russian	47.6	29.0	37.2
Chinese	47.5	22.2	35.2
Korean	47.7	23.3	35.6
Arabic	47.8	27.6	36.6
Hindi	47.5	26.0	31.7
Paren, real	47.5	10.5	35.0
Paren, synth	48.2	12.0	34.3
Disc, large	47.7	24.7	30.7
Disc, small	14.3	16.2	17.3
Rec, large	22.5	18.4	25.4
Yao+	4.0	20.1	25.6
Mu+, SW	3.3	18.4	23.3
Mu+, CUB	47.6	21.6	24.6
Random	1.8	3.0	19.7
No pretrain	11.4	4.3	28.1
Correl. with XferBench	−0.75	−0.84	−0.79

Table 4.2: chrF scores across three English-to-French machine translation settings. Correlation measured with the Pearson correlation coefficient. Colors normalized by column.

worse than the rest of the tested languages. *No pretrain*'s performance is worse than the cluster of synthetic and emergent languages but better than the fully random language (*Random*).

Machine Translation

The chrF scores of the machine translation experiment are given in Table 4.2 (BLEU scores in Section B.4). Additionally, we give Pearson correlation coefficients between each setting and the scores generated by XferBench (scatter plots shown in Section B.4). In all settings, we see that XferBench is strongly correlated with the results of the machine translation experiment.

For the *Full* setting, the results are somewhat inconclusive. Human languages perform the best and similarly to each other. *Paren, real*, *Paren, syn*, *Disc, large*, and *Mu+, CUB* all match the performance of human languages as well. The rest of the language perform significantly worse than the aforementioned languages, especially *Yao+* and *Mu+, SW* (see Section B.6 for sample outputs). In the case of *Random*, the training loss did not decrease during training likely due to the high learning rate.

In *Frozen*, we see the best correlation with the hypothesis regarding human languages (as well as with XferBench itself). *Disc, large* performs comparably to the worst human languages and better than the rest of the languages. The remainder of the synthetic and

emergent languages perform worse than the human languages but better than the random baselines.

Finally, *Reduced* (i.e., lower learning rate and tuning data) displays better separation than *Full*, but not as significant as *Frozen*. Human languages still perform the best, although they are matched by the *Paren* languages. *Disc, large* underperforms the human languages but still outperforms all other emergent languages. All emergent languages, apart from *Disc., large* underperform the *No pretrain* baseline. The better half of languages performed better (compared to themselves) with a higher learning rate while the lower half performed better with a reduced learning rate.

4.6 Discussion

Experiments

The basic ordering of the language by XferBench follows basic *a priori* assumptions: random baselines perform the worst, human languages perform the best, and emergent and synthetic languages are bounded above and below by these (supporting Hypothesis 1). Human languages cluster together in XferBench although there is still variation with non-overlapping confidence intervals (partially supporting Hypothesis 2).

Intra-EL differences Generally speaking, there is very little variation shown by XferBench on the emergent languages; nevertheless, we can still draw a handful of conclusions. First, *Disc, large* outperforms *Disc, small* while sharing the same codebase, task, etc. and differing only in message length, vocabulary size, observation space, and corpus size (supporting Hypothesis 3). This result matches the trend seen in Yao et al. [2022] that larger vocabularies and message lengths in an emergent language lead to better performance on downstream data. On the other hand, *Disc, small* performs similarly to other languages with larger vocabularies and longer message lengths (contradicting Hypothesis 3).

Second, it seems that the underlying complexity of the emergent communication game does not directly correlate with XferBench score: the abstract visual reasoning of *Mu+, SW* and *Mu+, CUB* does not lead to it outperform *Disc, small*. Additionally, the richer observations (i.e., image embeddings) of *Mu+, CUB* and *Yao+* also do not, by their mere presence, confer an advantage to the emergent language with respect to XferBench.

Finally, *Disc, large* and *Recon, large* both share hyperparameters in terms of the vocabulary size, message length, and corpus size, yet *Disc, large* shows significantly better performance on XferBench. This indicates that XferBench is not *solely* concerned with surface-level features as we see that the nature of the game (e.g., discrimination versus reconstruction, success rate) is relevant as well.

Correlation with MT The results from the machine translation experiment show strong, though not perfect, (negative) correlation with XferBench (supporting Hypothesis 4). For example, in all cases, *Disc, large* outperforms all other emergent languages. This strongly supports the notion that XferBench performance is predictive of downstream performance on more concrete NLP tasks.

The results from the *Full* setting of the MT experiment do show some correlation with XferBench but fail to show expected trends in other ways. For example, there is no clear ordering among the human languages (e.g., *French* does *not* outperform *Arabic*). Additionally *Yao+* and *Mu+, SW* drastically underperform the other emergent languages and the *No pretrain* baseline. We suspect that these aberrations from expected results come in part due to the high learning rate which cause unstable training or generation. On the other hand, the *Frozen* setting gives us the clearest ordering of human languages that matches with *a priori* expectations; this setting also has the strongest correlation with XferBench scores. The *Reduced* setting shows better correlation than *Full* but is not as clear as *Frozen*.

Random baselines In all of our experiments, the pretraining on random tokens (*Random*) performed notably worse than not pretraining at all (*No pretrain*), suggesting that ill-conditioning the neural network can be a significant hindrance to performing well on XferBench. This is important to note in light of the fact that a perfectly one-to-one compositional language describing uniformly sampled attribute–value vectors would yield a corpus with a uniformly random unigram distribution. This is to say, a fully compositional language, which is often seen as desirable in emergent communication research, could make for a very poor source of pretraining data as shown by *Random*'s performance on XferBench.

This fact along with the observations about sensitivity to learning rate indicates that performance on XferBench is not simply a function of the particular features of the emergent language in relation to the downstream human languages but also a function of the dynamics of optimization (i.e., priming the model to adapt well). Although this increases the difficulty of developing and interpreting a tool like XferBench, it is almost an unavoidable part of deep learning methods.

Future work

We identify three main directions for future work with XferBench. The first direction is determining what XferBench is measuring and how its scores correlate with the different factors of emergent languages. Yao et al. [2022, app. B.4] pursued this on a small scale with factors like vocabulary size and message length, but there exist a host of other factors worth exploring: speaker model size, game design, language entropy, observation modality, etc.

The second direction is more extensively investigating the correlation of XferBench with downstream tasks. We would expect that tasks that rely heavily on a language model—such as automatic speech recognition, abstractive summarization, and generative question-answering—to correlate well with XferBench. On the other hand, tasks that are more focused on classification—such as named entity recognition, sentiment analysis, and multiple choice question-answering—might not correlate as well.

Finally, XferBench would benefit greatly from improved compute efficiency. For example, if the results of XferBench could be replicated with a fraction of the training steps, it could (1) allow for a larger number of downstream languages to be tested which would reduce the size of the confidence intervals, allowing more more precise scoring. And (2), it would open the door to using larger models which would better capture the deeper structures of language and likely correlate better with realistic downstream tasks.

4.7 Conclusion

In this paper we have introduced XferBench, a first-of-its-kind benchmark for evaluating the quality of an emergent language corpus based on its transfer learning performance on human languages. This approach to evaluating emergent language scales with data and compute as opposed to requiring increasingly complex handcrafted rules to measure the desirable qualities of emergent language. We provide empirical results of XferBench across human, synthetic, and emergent languages and demonstrate that these results correlate with downstream performance on a machine translation task. XferBench is implemented as an easy-to-use Python package that will permit researchers in the field to easily apply XferBench to new emergent languages.

4.8 Limitations

The first limitation of XferBench is that it relies on a restricted interface with the emergent communication system. With emergent communication we have access not only to the grounding of all of the utterances of the emergent language but also full access to the agents themselves. Language is fundamentally a contextual phenomenon, so only a small part of it can be understood from looking at corpora in isolation. Thus, although XferBench is much more broadly applicable because of this restricted interface, it is also quite limited in what it can detect from a theoretical point of view.

The other set of limitations we will discuss have to do with the model and data size. First, the model and data size (60 M parameters and 15 M tokens) are quite small by contemporary standards, limiting the direct applicability of results from XferBench to relevant downstream tasks involving large language models, for example. On the other hand, scaling up the models, data, and methods of XferBench comes with its own difficulties. First, it would start to bias the benchmark towards high-resource languages, as only those could provide the necessary data to accommodate larger models. Second, it would make XferBench, which is already relatively slow as a metric (6 GPU-hours) even slower. This would decrease the speed of the iterative design process of emergent communication systems and, thus, the utility of the metric as a whole.

Chapter 5

Optimizing for Human Language Similarity with Transfer Learning [under review]

Abstract

In this chapter, 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 an emergent language as well as validate 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.[1]

5.1 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

[1]Based on "Searching for the Most Human-like Emergent Language" under review in the December 2024 cycle of the *Association for Computational Linguistics (ACL) Rolling Review*.

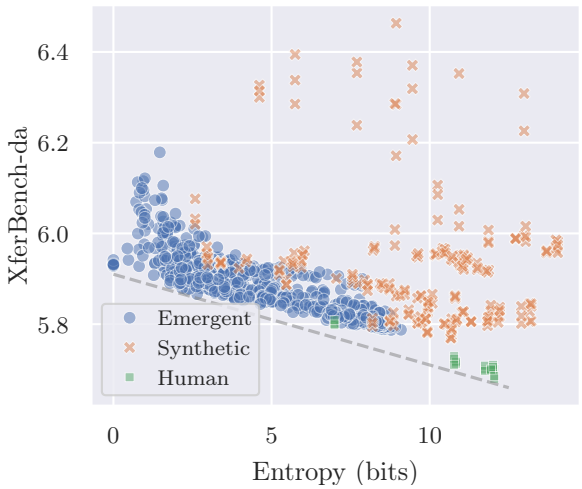


Figure 5.1: Hyperparameter search shows that emergent and human languages tend towards the Pareto frontier of minimizing entropy and minimizing XferBench score (lower is better) while non-emergent synthetic languages less reliably follow this trend. Dashed gray line represents a lower bound on entropy versus XferBench score.

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].

First and foremost, this moves the field of emergent language measurably closer to the goal of providing realistic, fully synthetic data for NLP. On a methodological level, hyperparameters in emergent communication research are often selected arbitrarily or based on convenience. Instead, hyperparameters ought to be selected, we suggest, such that they maximize emergent language's similarity to human language. 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 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.

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 Figure 5.1). 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 5.2. Methods are discussed in Section 5.3, and the experiments are presented in Section 5.4. An analysis of the results is performed in Section 5.5 with discussion and conclusion in Sections 5.6 and 5.7.

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 language, making them more statistically similar to human language; and (3) provide evidence that entropy minimization is a general property of emergent communication systems, showing that it is minimized with respect to transfer learning performance.

5.2 Related Work

For a general overview of deep learning-based emergent communication research, see Lazaridou and Baroni [2020]. This paper shares the goal of producing emergent language corpora that are suitable for transfer learning to human languages with Yao et al. [2022], which also introduces the *corpus transfer* method for applying emergent communication techniques to pretraining deep learning models used in this paper. Boldt and Mortensen [2023], similarly to this paper, investigate the effect of hyperparameters on emergent communication, although their study focuses primarily on the effects of individual hyperparameters on entropy instead optimizing an entire system 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 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.

5.3 Methods

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

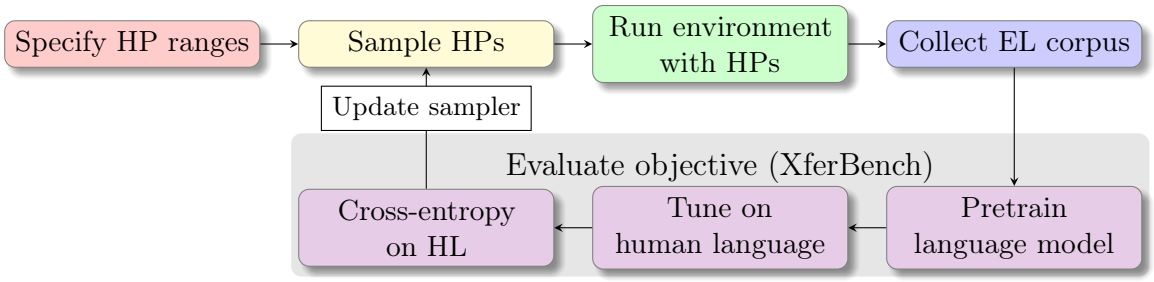


Figure 5.2: Illustration of hyperparameter optimization with XferBench (adapted from Boldt and Mortensen [2024b] (CC BY 4.0 License)).

task on human languages (illustrated in the gray box of Figure 5.2). 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 of the (human) evaluation 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 Section C.1 for details.

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., 2019, `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 one-hot vectors.
- 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.
- 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 "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 and feeding these observations into the sender which generates the message.

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:

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 Section C.2.

Optimization: hyperparameter search

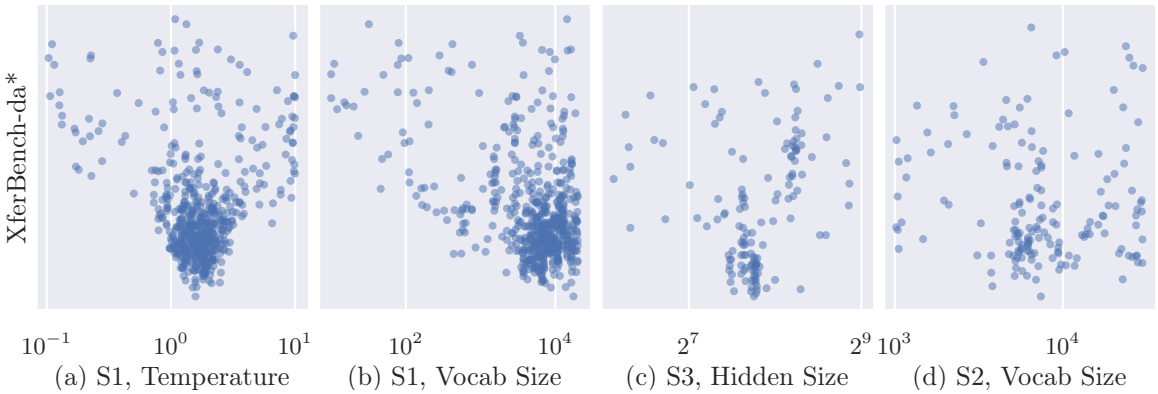
Finally, we discuss the method used for optimizing the hyperparameters of the emergent communication system (the parameters system itself are optimized with backpropagation, as mentioned above). The simplest of all hyperparameter search methods is grid search, where each element of the Cartesian product 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

[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 5.1: 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.



(a) S1, Temperature (b) S1, Vocab Size (c) S3, Hidden Size (d) S2, Vocab Size

Figure 5.3: 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.

us to specify a range of hyperparameter values instead of individual values. This process is illustrated in Figure 5.2.

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].

5.4 Experiments

The code to run the experiments and analyses is publicly available at [supplementary material for review] under the MIT license.

Hyperparameter searches

In this paper, we present four main searches (Searches 1–4, parameters given in Table 5.1) with two additional searches (Searches 5r and 6e) for use in later analyses (Section 5.5). 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 5.1 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 5.5.

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 5.5.

The parameters of Searches 1–4 are given in Table 5.1 (for complete table, see Table C.2). 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 Section C.4 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 patterns[4], illustrated in Figure 5.3. For parameters with a clear trend toward the center (Figure 5.3a), we narrowed the range to encourage exploiting good values. Some parameters trended to one side of the range (Figure 5.3b), which indicated needing to extend the range. Parameters with weak to no trend (Figures 5.3c and 5.3d) were left unchanged for the initial searches and given an arbitrary value for the final search to reduce additional noise. Full hyperparameter plots given in Section C.7.

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

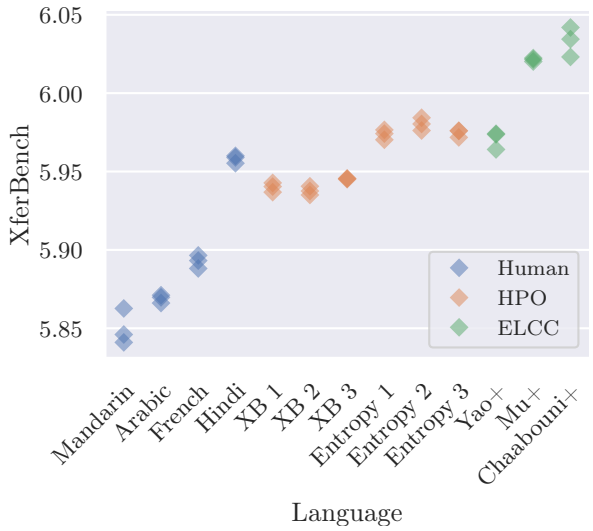


Figure 5.4: Bar chart of XferBench scores on emergent and human languages. XB 1–3 are emergent language corpora derived from Search 4 and Entropy 1–3 from Search 6e.

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 5.5).

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, 2021b]) and Chaabouni+ (`ec-at-scale/imagenet-10x10` [Chaabouni et al., 2022]) were also included as more typical high-performing emergent languages on XferBench.

Results

Figure 5.4 shows 3 randomly seeded runs of the full XferBench score for each corpus. For the emergent languages from hyperparameter search, the models 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. 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.

5.5 Analysis

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 Figure C.5); 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 than are often used 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.

Entropy and XferBench

The most striking correlation we observe in our experiments is between XferBench score and unigram token entropy, which is illustrated in Figure 5.1 (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 Figure 5.1). 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 low entropy reduction (higher cross-entropy) language models.

Entropy minimization Looking again at Figure 5.1, 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 Section C.5) 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 briefly in Section C.6).

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 Figure 5.4).

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. 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 has a high entropy.

Task success The correlation between task success and XferBench score (Figure 5.5, Pearson's $r = -0.40$) is not as dramatic as with entropy. Nevertheless, the negative

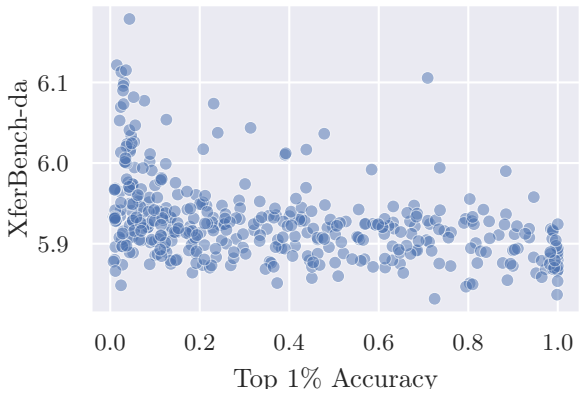


Figure 5.5: Accuracy versus XferBench for Search 5r. Accuracy is measured as proportion of rounds for which the correct observation is ranked in the top-1 percentile among all distractors.

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.

5.6 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]. 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 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.

5.7 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. 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.

Regrading 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.

Chapter 6

Discovering Morphemes in Rich Corpora [proposed]

Abstract

We introduce CSAR, an algorithm for inducing morphemes from emergent language corpora of parallel utterances and meanings. It is a greedy algorithm that (1) weights morphemes based on mutual information between forms and meanings, (2) selects the highest-weighted pair, (3) removes it from the corpus, and (4) repeats the process to induce further morphemes (i.e., Count, Select, Ablate, Repeat). The effectiveness of CSAR is first validated on procedurally generated datasets and compared against baselines for related tasks. Second, we validate CSAR's performance on human language data to show that the algorithm makes reasonable predictions in adjacent domains. Finally, we analyze a handful of emergent languages, quantifying linguistic characteristics like degree of synonymy and polysemy.

6.1 Introduction

Emergent languages—communication systems invented by neural networks via reinforcement learning—are fascinating entities. They give us a chance to experiment with the processes underlying the development of human language to which we would not otherwise have access. A perennial problem in this field, though, is that emergent languages are difficult to interpret. The strategies emergent languages use to convey meaning do not always align with those known from human language [Kottur et al., 2017, Chaabouni et al., 2019b, Kharitonov and Baroni, 2020b]. Yet a lack of general-purpose methods for investigating the structure of emergent communication prevents us from systematically investigating how they encode meaning.

As an essential step towards understanding emergent languages, we introduce CSAR, an algorithm for *morpheme induction* on emergent language. That is, given an input corpus of parallel data: utterances and their associated meanings, find the smallest meaningful components of utterances with their accompanying meaning. Simply put, this task is to jointly segment utterances and align them with their meanings. The output of this algorithm, then, is a mapping between the forms and meanings of the emergent language (example

Form	Meaning
3, 6	{not, gray}
7, 7	{not, blue}
32	{circle}
4, 5	{not, yellow}
6, 12, 6, 12	{green, or, yellow}
3, 12, 3	{blue, or, yellow}

Figure 6.1: Example of morphemes extracted from a signalling game with pixel observations.

shown in Fig. 6.1). Furthermore, the proposed algorithm is easily applicable to almost any emergent language due to the simplicity of the input format. In fact, the algorithm is general purpose enough to produce reasonable results in other domains, as we demonstrate with human language-based image captioning, machine translation, and word segmentation data. Validating CSAR's performance on human language data as well as synthetic data is critical to demonstrating its effectiveness since there is no way of obtaining ground truth morphemes for emergent languages.

An inventory of the morphemes of an emergent language is the foundation of many further linguistic analyses. Existing studies of compositionality [Korbak et al., 2020], word boundaries [Ueda et al., 2023], and grammar induction [van der Wal et al., 2020] could be validated and augmented with information on the morphology of emergent languages, and new directions would also be made possible, including analyses of the morphosyntactic patterns and typological properties of emergent languages. Ultimately, such studies form one of the pillars of emergent communication research: learning what emergent language can tell us about human language [Boldt and Mortensen, 2024c].

In Section 6.2 we define the task of morpheme induction and discuss related work. Section 6.3 presents the proposed algorithm, CSAR. Section 6.4 validates CSAR's performance on procedurally generated and human language data. Section 6.5 applies CSAR to emergent languages. And we discuss the paper's findings and limitations and conclude in Sections 6.6 to 6.8.

Contributions This work: (1) Introduces an algorithm for inducing morphemes, applicable to a wide variety of emergent language corpora. (2) Offers an easy-to-use Python implementation for executing the proposed algorithm on arbitrary emergent language data. (3) Provides a first look into the morphology of emergent languages including phenomena such as polysemy and synonymy.

6.2 Problem Definition

In this section we give a precise definition of the problem and the terms we will use throughout the paper.

Task: morpheme induction

We define the task of *morpheme induction* as follows: Given a corpus of *utterances* and their corresponding *complete meanings*, identify *minimal, well-founded form–meaning pairs* (i.e., *morphemes*) present in the corpus. This collection of pairs is the *morpheme inventory* of the corpus.

form a sequence of (form) tokens; represented as an integer sequence in emergent language.
utterance a complete sequence of tokens produced by an agent; forms are subsequences of utterances.
meaning a set of meaning tokens (i.e., atomic meanings grounded in the environment).
complete meaning a meaning which represents the entire meaning of an utterance.[1]
well-founded a form–meaning pair is well-founded when the particular form corresponds with a particular meaning.
minimal a well-founded form–meaning pair is minimal when there is no way to decompose the pair while maintaining continuity of meanings.

It is important to note that we make two assumptions about the complete meanings. First, complete meanings are assumed to be *abstracted* already, hence the reason we can represent them as a set of atomic meanings. That is to say that the "raw semantics" of the utterances are already broken down into individual components of interest; this task does not entail automatically finding meaning in arbitrary data (cf. clustering). Second, since complete meanings are sets, they are not able to represent more complex phenomena that might require graph structures, for example (cf. abstract meaning representations).

Additionally, we note that *well-founded correspondence* is a concept subject to a variety of philosophical accounts. Sometimes these accounts hold that the meaning is derived from either the behavior or state of mind of a language user. Yet in this task, we only have access to a corpus, not to the language users themselves; thus, we employ a notion of "well-founded correspondence" most akin to a statistical view semantics (e.g., as in the distributional hypothesis).

Related work

Emergent language Lipinski et al. [2024] serves as the inspiration for this paper through its application of normalized pointwise mutual information to probe emergent languages for certain kinds of form–meaning relationships, though it stops short of providing full morpheme inventories over arbitrary data. Ueda et al. [2023] introduces a method of form-only segment induction for emergent language based on token-level entropy patterns in utterances.

Finally, Brighton [2003a] introduce methods for inducing morphemes from simulations of language evolution. In particular, the algorithm is based on finite state transducers and the minimum description length principle. The key difference, though, is that the FST-based method assumes a strict form–meaning correspondence that does not appear to hold in emergent languages generated by deep neural networks. Thus, CSAR is designed to handle noise and looser form–meaning correspondence.

[1] atomic meaning $\in$ meaning $\subseteq$ complete meaning

Statistical word alignment The task of morpheme induction resembles the task of statistical word alignment for machine translation insofar as it involves learning a mapping between two modalities. Well-known algorithms for this task include the IBM alignment models [Brown et al., 1993]. While morphemes can be extracted from the alignments, the alignments themselves are not intended to represent morphemes as such.

Segment induction Segment induction is similar to morpheme induction, except that it deals only with the forms; these methods, then, cannot provide a mapping between form meaning because they are meaning-unaware. Sometimes this task is called "morpheme induction" since the segments are supposed to correspond to morphemes, but they are not morphemes in the particular sense we use for this paper, that is: explicit form–meaning pairs. An example of an algorithm which addresses this task is Morfessor [Creutz and Lagus, 2002, Virpioja et al., 2013] or the submissions to the SIGMORPHON 2022 Shared Task Batsuren et al. [2022]. Narasimhan et al. [2015] introduce a semi-supervised segment induction algorithm that uses semantic features to guide segmentation (viz. groups of morphologically related words), although meanings are not modelled explicitly and "word" is not a well-defined concept for emergent languages. The discovery of valid segments by tokenization methods based on statistics—such as BPE [Sennrich et al., 2016, Gage, 1994b] and Unigram LM [Kudo, 2018]—is largely an epiphenomenon, not a design goal.

6.3 Algorithm

In this section we introduce the algorithm for morpheme induction: CSAR (Count, Select, Ablate, Repeat). CSAR comprises the following steps:

1. Collect form and meaning candidates from the corpus.
2. While form and meaning candidates remain.
   a) Count co-occurrences of form and meaning candidates.
   b) Select form–meaning pair with the highest weight.
   c) Remove instances of the form–meaning pair from the corpus.
3. Selected form–meaning pairs constitute the morpheme inventory of the corpus.

The code implementing CSAR as well as the experiments discussed later is available under a free and open source license at `https://github.com/brendon-boldt/csar`.

Representation and preprocessing

Input data The input data to CSAR is a parallel *corpus* of utterances and their meanings. Each record in the corpus is a tuple of form and meaning where a form is a list of (form) tokens and a meaning is a set of (meaning) tokens.

Candidate collection Given the corpus, we can identify and count the *form* and *meaning candidates* to produce their corresponding *occurrence matrices*. A form candidate is any substring of form tokens under consideration for inducing morphemes. A meaning candidate is any subset of meaning tokens under consideration for inducing morphemes. The most

straightforward approach is to simply consider every non-empty substring of forms and subset meanings, although CSAR is not constrained to this approach in theory (cf. Section D.1).

Having defined the universe of forms and meanings, we can build a binary *occurrence matrix* for forms and one for meanings, where each row corresponds to a record and each entry corresponds to the presence (1) or absence (0) of a form/meaning in that record. Thus, the form occurrence matrix has the shape $O_{\mathcal{F}} : |\mathcal{R}| \times |\mathcal{F}|$ and the meaning matrix $O_{\mathcal{M}} : |\mathcal{R}| \times |\mathcal{M}|$, where $\mathcal{R}$ is the list of records, $\mathcal{F}$ is the set of all forms candidates, and $\mathcal{M}$ is the set of all meanings candidates.

Example If we had a simple corpus with records ("s", □), ("st", ⊠), ("ct", ⊗), the corresponding occurrence matrices would be:

$$\mathcal{O}_{\mathcal{F}} = \begin{bmatrix} \cdot & \mathrm{s} & \cdot & \cdot & \cdot \\ \cdot & \mathrm{s} & \mathrm{t} & \cdot & \mathrm{st} \\ \mathrm{c} & \cdot & \mathrm{t} & \mathrm{ct} & \cdot \end{bmatrix} \quad \mathcal{O}_{\mathcal{M}} = \begin{bmatrix} \square & \cdot & \cdot & \cdot & \cdot \\ \square & \times & \cdot & \boxtimes & \cdot \\ \cdot & \times & \bigcirc & \cdot & \otimes \end{bmatrix}, \tag{6.1}$$

where entries with value 1's are shown with the occurring symbols, and entries with value 0's with · for clarity.

Main loop

Weighting and co-occurrences Given the occurrence matrices, the next step is to compute the weights of all potential pairs. The pair with the highest weight will be selected and added to the morpheme inventory. The weight of a form–meaning pair is the mutual information of the binary variables representing the corresponding form and meaning. The mutual information of a particular form–meaning pair is given by

$$I(F; M) = \sum_{x \in F} \sum_{y \in M} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)}, \tag{6.2}$$

where $F = \{f, \neg f\}$, $p(f)$ is the probability of f appearing in a record, $p(\neg f)$ is the probability of f not appearing, and the rest are defined analogously. The key term of the mutual information expression is the joint probability between a form and a meaning, $p(f, m)$: since f and m are binary variables, all other joint probabilities can be computed from their joint probability and the marginal probabilities. The joint probability can be computed by normalizing the sum of co-occurrences of given forms and meanings, namely:

$$p(f, m) = \frac{1}{|\mathcal{R}|} \sum_{j=1}^{|\mathcal{R}|} O_{\mathcal{F}}[j, i_f] \wedge O_{\mathcal{M}}[j, i_m] \tag{6.3}$$

where i_f and i_m are the indices of f and m in their respective matrices. More succinctly, co-occurrences can be computed with matrix multiplications, yielding

$$p(f, m) = \frac{1}{|\mathcal{R}|} \cdot \left(O_{\mathcal{F}}^{\top} O_{\mathcal{M}}\right)[i_f, i_m] \tag{6.4}$$

Other weighting methods were explored including joint probabilities, pointwise mutual information, and normalized pointwise mutual information, though mutual information was found to perform best empirically.

The above weighting function results in ties which we break with the following heuristics: (1) higher initial weight, (2) fewer selected pairs with this form, (3) larger form size, and (4) smaller meaning size.

Remove pair from corpus The final step of the algorithm's main loop is ablating the pair from the corpus. That is, once we select a form–meaning pair, we want to remove all co-occurrences of the form and meaning in order to determine what form–meaning correspondences remain to be explained. For example, after ablating the pair ("t", ×), the corpus from above would comprise ("s", □), ("s", □), and ("c", ○); the occurrence matrices would then be updated to reflect this. In cases where ablating a pair is ambiguous, we apply a heuristic (see Section D.1).

Repeating and stopping After ablating the selected form–meaning pair, the algorithm repeats the main loop, beginning again at the weight-computation step (with the updated occurrence matrices). The one difference is that—in subsequent weight computations—the weight of a pair cannot go up, preventing spurious correlations from arising in later steps.

This loop continues until form or meaning occurrences are exhausted or some other criterion is met (e.g., time limit, inventory size limit). In this way, CSAR is an "anytime" algorithm since it can be stopped after an arbitrary number of iterations and still produce a sensible result. This is because the most heavily weighted morphemes can be considered the highest *confidence* morphemes, meaning that stopping the algorithm before completion will only leave out the lowest confidence morphemes.

Implementation

The implementation of CSAR introduced in this paper is written in Python making use of sparse matrices from `scipy` [Virtanen et al., 2020, BSD 3-Clause license] and JIT compilation with `numba` [Lam et al., 2015, BSD 2-Clause license] to speed up execution. CSAR is conceptually simple. Most of the implementation complexity lies in efficiently handling the occurrence matrices, especially when removing a form–meaning pair from the corpus. For example, the co-occurrence matrix has the shape $|\mathcal{F}| \times |\mathcal{M}|$ which is massive considering that $\mathcal{F}$ and $\mathcal{M}$ are already accounting for the universes of all possible forms and meanings in the corpus. Nevertheless, there are a wide range of heuristics that can be applied to greatly speed up execution while maintaining performance (see Section D.1).

6.4 Empirical Validation

To validate the ability of CSAR to find well-founded morpheme inventories, we test it against procedurally generated datasets as well as human languages. Since we do not have access to ground truth morphemes for emergent languages, we gauge the effectiveness of CSAR's morpheme induction in the next best way: by testing its performance in these adjacent domains. Procedurally generated datasets (described in Section 6.4) both give us access to the "ground truth" morphemes and allow us to vary particular facets of the languages. Having ground truth morphemes allows us to quantitatively evaluate CSAR against baseline methods

(Section 6.4). Fine-grained control over the facets of the languages permits us to identify particular phenomena that are challenging for CSAR to induce correctly (Section 6.4). We also test CSAR against human language data (Section 6.4) in order to give a qualitative sense of the effectiveness of the algorithm.

Procedural datasets

The dataset-generating procedure has the following basic structure: (1) Meanings are sampled according to some structure (viz. a fixed attribute–value vector). (2) An utterance is produced from this meaning according to a mapping of meaning components to form components. (3) The form–meaning pairs that were used to generate the utterance are added to the set of ground truth morphemes. In the basic case, for example, each particular attribute and value is associated with a unique sequence of tokens which are concatenated to form an utterance, creating a one-to-one mapping from meanings to forms.

Variations Such languages are trivial to induce morphemes from, so we introduce the following variations to produce more complex datasets:

Synonymy Multiple forms may correspond to the same meaning.
Polysemy Multiple meanings may correspond to the same form.
Multi-token forms A form may comprise more than one token, possibly overlapping with other forms.
Vocab size Number of unique tokens.
Sparse meanings Meanings occur independently of each other with no additional structure (i.e., not structured as attribute–value pairs).
Distribution imbalance Meanings are sampled from non-uniform distributions.
Dataset size Number of records in the dataset.
Number of meanings Total number of meanings (e.g., varying number of attributes and values).
Noise forms Form tokens not corresponding to any meanings are added.
Shuffle form Inter-form order is varied randomly (while maintaining intra-form order).
Non-compositionality A given form may correspond to multiple meanings simultaneously.

For the following analyses, we report values for a collection of procedural datasets built from the Cartesian product of two values for each of the above variations (one where the variation is inactive and one where it is). See Section D.2 for details.

Evaluation metric We use F_1 score (harmonic mean of precision and recall) to assess the quality of an induced morpheme inventory given the ground truth inventory. We define precision as

$$\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\max_{g\in\mathcal{G}} s(i,g), \tag{6.5}$$

where $\mathcal{I}$ is the set of induced morphemes, $\mathcal{G}$ is the set of ground truth morphemes, and s is the similarity function. For exact F_1, the similarity function is 1 if the morphemes

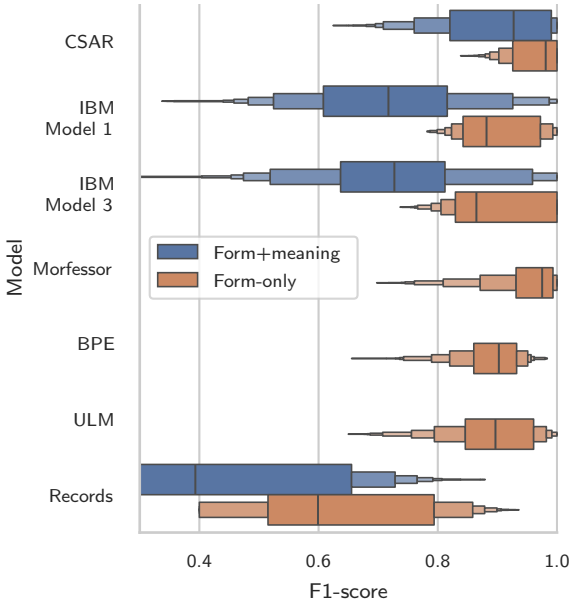


Figure 6.2: Fuzzy F_1 scores for CSAR and baseline methods across procedural datasets. Results reported for form–meaning inventories and form-only inventories.

are identical and 0 otherwise. In fuzzy F_1, the similarity function is the minimum of form similarity (normalized insertion–deletion ratio[2]) and meaning similarity (Jaccard index). Recall is defined similarly to precision except that the roles of $\mathcal{I}$ and $\mathcal{G}$ from Eq. (6.5) are reversed.

Comparison with baselines

Below we describe the baseline methods we use for comparison.

IBM Model 1 Simple expectation-maximization approach to machine translation primarily through aligning words in a sentence-parallel corpus. [Brown et al., 1993]

IBM Model 3 Built on top of the IBM Model 1 to handle phenomena such as allowing a form to align to no meaning.

Morfessor A form-only segmentation algorithm built to handle human language; also uses an EM algorithm.

Byte pair encoding A greedy form-only tokenization method which recursively merges frequently occurring pairs of tokens. Vocabulary size is selected according to a simple heuristic (see Section D.2). [Gage, 1994b, Sennrich et al., 2016]

Unigram LM An EM-based form-only tokenization method which starts with a large vocabulary and iteratively removes tokens contributing least to the likelihood of the data. Vocabulary size is selected according to a simple heuristic (see Section D.2). [Kudo, 2018]

Record A trivial baseline where the inventory is just the set of all records.

[2] $1 - (\text{insertions} + \text{deletions})/(|s_1| + |s_2|)$

Dataset	Induced Morpheme
Morphology	(“ed$”, {PAST}) (“ ’ ”, {POSSESSIVE})
Image captions	(“stop sign”, {stop sign}) (“woman”, {person}) (“skier”, {person, skis})
Translation	(“Member States”, {Mitgliedstaaten})

Figure 6.3: Examples of morphemes induced from various human language datasets and tasks.

For the baseline methods which do not handle meanings and only produce forms, we report the form-only F_1 score (i.e., $s(i, g)$ only takes the form into account), though CSAR and IBM models still have access to meanings. For form-only metrics, we exclude datasets which include noise forms as form-only methods cannot identify which forms are noise.

Results The results of CSAR and the baselines on the procedural datasets are presented in Fig. 6.2, which shows the distributions of mean scores for each hyperparameter setting for the procedural datasets. Each setting was repeated over 3 random seeds. Additional results are given in Section D.2. For inducing full morphemes (form and meaning), CSAR performs the best by a large margin over the baselines (and even greater margin when considering exact F_1). The IBM alignment models perform better than the trivial record-based baseline but still perform noticeably worse than CSAR. While CSAR yields roughly equal precision and recall, the IBM models' precision is lower than their recall suggesting that they are more prone to inducing spurious morphemes than CSAR.

When evaluating the forms only, we find that CSAR is the best method with Morfessor exhibiting comparable performance. The IBM alignment models exhibit roughly the same performance as the tokenization methods (BPE and Unigram LM). As with the full morpheme results, CSAR is the only method to achieve comparable precision and recall with all other baselines having precisions lower than their recalls.

Error Analysis

For the most part, the errors CSAR makes are “edit errors”: identifying a correct morpheme but adding or removing a form or meaning token. This is reflected in the near-parity between precision and recall. This is in contrast to the baseline methods which are more prone to inducing too many morphemes, leading to lower precision.

Generally speaking, as more variations are added to a dataset, the performance degrades further. In particular, CSAR's performance decreases the most with small corpus sizes, overlapping multi-token forms, and non-compositional mappings. On the other hand, using sparse meanings, shuffling the forms, and using a non-uniform meaning distribution have relatively little effect.

Human language data

In this section we discuss the results of running CSAR on three different human language datasets. While these datasets are not the intended domain of CSAR—and CSAR is certainly not the best algorithm for the tasks—the point of these experiments is to demonstrate the general effectiveness of the algorithm qualitatively (examples shown in Fig. 6.3). Since these datasets are larger, we employed heuristic optimizations to CSAR to reduce their runtime (described in Section D.1). The top 100 induced morphemes for each human language dataset are given in Section D.5.

Morpho Challenge The first human language dataset we use is from the Morpho Challenge [Kurimo et al., 2010]. This dataset is a human language approximation of the task of morpheme induction for emergent language. Concretely, the utterances are single English words, divided up at the character level, while the meanings are the constituent morphemes.

CSAR is able to recover a wide variety of morphemes including: roots like ("^fire", {fire}), prefixes like ("^re", {re-}), suffixes like ("ed$", {PAST}), and other affixes like (" '", {POSSESSIVE}). While the vast majority of morphemes CSAR induces are accurate, a handful of the lowest-weighted morphemes are spurious (e.g., ("s$", {boy})) likely due to inaccurate decoding earlier in the process (i.e., part of the true form for a given meaning was included in a prior meaning).

Image captions The next dataset we employ is the MS COCO dataset [Lin et al., 2015, CC BY 4.0]. In particular, we take the image captions to be the utterances, treating words as atomic units, and the meaning to be the labeled objects in the image (e.g., person, cat).

The bulk of highest weighted induced morphemes are direct equivalents of the objects they describe (e.g., ("cat", {cat})). We find instances of synonymy (e.g., ("bicycle", {bicycle}) and ("bike", {bicycle})) as well as polysemy (e.g., ("animals", {cow}) and ("animals", {sheep})). Finally, we also observe compound forms like ("stop sign", {stop sign}) as well as compound meanings such as ("skier", {person, skis}). As we go beyond the top 100 or so, the associations between forms and meanings remain reasonable but become looser such as ("bride", {dining table, tie}) or ("sink", {toothbrush}).

Machine translation For machine translation, we use the WMT16 dataset and the English–German split, in particular [Bojar et al., 2016]. In this case, the English text is considered to be the utterance and the German text to be the meaning, with words being the atomic units on both sides.

As with the image caption results, the bulk of induced morphemes are direct equivalents (e.g., ("and", {und})). Beyond these simple one-to-one mappings, CSAR induces the polysemic relationship ("the", {der}) and ("the", {die}). Finally, CSAR also picks up on multi-token forms like ("Member States", {Mitgliedstaaten}).

6.5 Analysis of Emergent Languages

Here we apply CSAR to a handful of emergent language environments and observe various metrics which can be derived from the induced morpheme inventories. Although we do not have ground-truth morphemes for these environments, we can still observe patterns in the induced morphemes which demonstrate the diagnostic power of CSAR.

Environments

The selected emergent language environments span three different observation spaces: one-hot vectors, synthetic images of shapes, and natural images of birds. Within each of the environments, further signalling game variations are described below.

Vector observations In the vector observation signalling game (based on the EGG framework [Kharitonov et al., 2021]) the agents directly observe one-hot vectors without having to further extract representations. Specifically, we use two variants of the observation space: (1) the standard attribute–value setting where each of 4 attributes can take on 4 distinct values and (2) the "sparse" setting where there are 8 binary attributes which are either present or absent. During the training of the emergent language agents, the sparse variant is identical to the attribute-value variant (except for the number of attributes and values), but when the corpora are passed to CSAR, the attributes with a value of 0 are filtered out, yielding variable length meanings. Hyperparameters for both environments are given in Section D.3.

For both observation spaces, we also test two variations of the signalling game. The first is the *discrimination game*, where the receiver must answer a multiple-choice question where the correct observation is accompanied by multiple (incorrect) distractors. The second is the *reconstruction game*, where the receiver must recreate the original observation similar to the decoding segment of an autoencoder.

Image observations The second environment is introduced by Mu and Goodman [2021a] and uses both synthetic and natural images for its observation space. The synthetic images come from ShapeWorld, a tool for generating images of shapes with varying properties [Kuhnle and Copestake, 2017a] while the natural images are from the CUB-200-2011 dataset documenting various kinds of birds [Wah et al., 2011]. The meanings for these environments are discrete attributes describing the images (e.g., "red" for ShapeWorld or "wing color: brown" for CUB) provided with the datasets. In addition to a more standard discrimination game (termed *reference game* in this paper), Mu and Goodman [2021a] also introduce the *set-reference* and *concept* variants of the discrimination game. These variations increase the level of abstraction in the game in order to encourage more generalizable (and compositional) languages to emerge.

Specifically, the reference game functions like the discrimination game described above except that there are multiple target images and the sender sees both the target images and the distractors. The receiver, in this case, must classify the each image as being target or distractor (i.e., multi-label instead of multi-class). In the reference game, the target images are all identical while in the set-reference game, the target images are variations of the same

object (e.g., a red triangle in different positions/rotations). Finally, in the concept game, the target images comprise different objects sharing one or more concepts (e.g., triangular).

Metrics

We present a handful of quantitative metrics to act as summary statistics for the morpheme inventories induced from the above emergent languages.

Inventory entropy Entropy (in bits) of the morphemes according to their prevalence (probability of occurring out of all morpheme occurrences identified during induction). We use the inventory entropy to give a rough sense of the breadth of the morpheme inventory as CSAR can induce a large number of low-prevalence, low-quality morphemes (whereas entropy is more robust to this). A rough translation to size is $H^2 \approx$ equivalent number of equally-probable morphemes.

Morpheme bijectivity Weighted mean normalized pointwise mutual information (NPMI) of morphemes in the inventory corresponding to how "one-to-one" the morphemes identified are; higher bijectivity corresponds to higher quality morphemes/inventory. See [:reference phrasebook chapter] for the introduction of this metric.

Topographic similarity Correlation (Spearman's ρ) between distances in the utterance space and complete meaning space [Brighton and Kirby, 2006b, Lazaridou et al., 2018a] of the corpus (i.e., not based on the morpheme inventory).

Synonymy Entropy across forms for a given meaning; computed using prevalence of forms normalized within the particular meaning.

Polysemy Entropy across meanings for a given form; as with synonymy, *mutatis mutandis*.

Form size Mean number of tokens in a form, weighted by morpheme prevalence.

Meaning size Mean number of tokens in a meaning, weighted by morpheme prevalence.

With the exception of inventory size and toposim, the above metrics are weighted by *prevalence* which is the proportion of records from which the morpheme was ablated.

Results

The morpheme inventory metrics are given in Table 6.1 where they averaged across 10 runs of each environment; some examples of morphemes from the inventories are given in Section D.5.

Inventory, form, and meaning size Inventory entropy along with form and meaning size give sense of the general "shape" of the morpheme inventories—how large they are as a whole and what their components look like. With respect to inventory entropy, we find that generally, the image-observation environments results in higher entropy morphemes, which is to be expected with richer input source. On the other end of the interval, the (Vector, Attribute–value, Reconstruction) environment shows the lowest almost matching the lower bound of 4 bits.[3]

[3]Since this environment as 4 attributes which can each taken on 4 values, that would correspond to $4 \cdot 4 = 16$ unique, equiprobable morphemes (order-dependence notwithstanding), corresponding to $\log_2 16 = 4$ bits of entropy.

Obs Type	Subtype	Game Type	Inventory Ent.	Form Length	Meaning Size	Synonymy	Polysemy	Bijectivity	Toposim
Vector	Attr–val	Discrim	6.2	2.2	1.2	1.5	0.6	0.45	0.46
Vector	Attr–val	Recon	4.2	1.2	1.0	0.2	0.1	0.92	0.84
Vector	Sparse	Discrim	5.9	1.8	1.3	2.0	0.9	0.34	0.44
Vector	Sparse	Recon	5.4	1.3	1.2	1.7	0.8	0.44	0.56
Image	Synth	Ref	4.5	1.6	1.1	1.2	0.7	0.07	0.00
Image	Synth	Set-ref	6.4	1.2	1.7	0.8	2.2	0.28	0.12
Image	Synth	Concept	6.2	1.3	1.4	1.2	2.0	0.25	0.12
Image	Natural	Ref	7.1	1.1	1.9	0.3	3.6	0.09	0.05
Image	Natural	Set-ref	6.9	1.1	1.9	0.3	3.2	0.25	0.30
Image	Natural	Concept	6.9	1.2	2.2	0.2	2.8	0.47	0.34

Table 6.1: Metrics (described in Section 6.5) across the morpheme inventories of various emergent languages (averaged across 10 runs).

While both form length and meaning size remain small, their is a notable presence of sizes greater than 1. For form length, this suggests that the assumption that individual form tokens can be treated as words is not well founded; instead, the smallest meaningful units can comprise multiple tokens. For meanings, this indicates some degree of "fusionality" where the minimal unit of meaning corresponds to multiple atomic meanings at once and inseparably.

Inventory quality Insofar as morpheme bijectivity measures the quality of the morpheme inventories induced, we find that the quality across most environments is medium-to-low with only the reconstruction game with attribute–value vector observations averages above 0.5 [:Reference the phrasebook plot that gives a general sense of how well things go given a bijectivity.]. Generally speaking, the bijectivity values for the vector-based environments are higher than the those of the image-based environments, suggesting that the noiseless, pre-disentangled observations result in languages which are more easily captured by CSAR. The reference games in the image-based environments show especially low bijectivity potentially, in part, because the communication can reference low-level observation features without significant pressure towards generality.

Synonymy and polysemy Synonymy and polysemy, taken together, naturally correlate with morpheme bijectivity as the former metrics test for unidirectional correlation (e.g., how informative is a form for various meanings) while NPMI (the basis for bijectivity) takes bidirectional correlation (i.e., do a form and meaning correlate with each other exclusively). Empirically, the vector-based environments have more synonymy than polysemy while the image-based have this pattern reversed. Our initial hypothesis would be that synonymy would exceed polysemy insofar as polysemy incurs a cost with respect to communicative efficacy (i.e., ambiguity) while synonymy does not. That being said, the fuzzier relationship between the observations and the semantics in image environments (since it is a lossy conversion) compared to the one-to-one relationship between meanings and observations in the vector-based environments may explain part of the unexpected behavior.

Compositionality We find that morpheme bijectivity and toposim are relatively well correlated in our quantitative analysis, as is expected since they both aim to be measures of compositionality. On the other hand, we do not find any notable correlation between bijectivity/toposim and meaning size which could also be viewed a measure of compositionality; viz. fewer meaning components per morpheme correspond with a more one-to-one relationship between forms and atomic meanings (cf. holistic languages with many meaning components per form). The bijectivity–toposim correlation is also reflected in the progression from reference to set-reference to concept games in the image-based environments: not only does the toposim generally increase moving along the game progression (as presented in Mu and Goodman [2021a]) but the morpheme bijectivity as well. This gives even stronger evidence for claims of Mu and Goodman [2021a] as morpheme bijectivity is argued to be a better measure of compositionality that toposim [:ref later chapter].

6.6 Discussion

Due to CSAR's strong performance and easy application to a wide variety of emergent language environments, it would be a valuable addition to the standard toolkit of emergent language analyses. In particular, it helps fill a gap of environment-agnostic methods for interpreting the ways that emergent languages convey meaning—a perennial question in the field. Down the road, this opens up research questions concerning the evolution of meaning in emergent language, such as those discussed in Brighton [2003a], but with the ability to deal with the larger scale and particular difficulties of *deep learning-based* emergent communication.

Furthermore, morpheme inventories are a foundation for higher-level linguistic analyses of emergent language like inducing their syntactic structure. To skip the morpheme induction step would be comparable to attempting to understand the grammatical role of the letter *C* in English. Such analyses of the syntax of emergent language and beyond are critical to understanding how emergent and human language are similar and how they are different.

6.7 Conclusion

CSAR presents a strong platform for investigating the morphology of emergent language, demonstrating the ability to find minimal form–meaning pairs in both procedural and human language data. Given the morpheme inventory of an emergent languages we can not only analyze phenomena like synonymy and polysemy but also the typological features of emergent languages, determining which human languages they most closely resemble, if they resemble any. Such a study of morphology forms the foundation for the more general study of the linguistic features of emergent language and unlocks the door to the insights they can provide us about human language.

6.8 Limitations

Greed is not always good While the greediness of CSAR does simplify induction (conceptually and implementation-wise), improve runtime, and provide good partial inventories, it suffers from the same limitation inherent to greedy algorithms: it can get trapped in local optima. For example, it is possible to construct corpora for which a greedy approach is "misled" since certain heuristics require revision based on information encountered later in the induction process (e.g., preferring smaller versus larger forms).

We did consider non-greedy approaches to morpheme induction but ultimately decided not to pursue them in this work because (1) the greedy approach itself demonstrated strong performance and (2) initial attempts at non-greedy approaches (e.g., tree search) yielded intractable runtimes. For example, an error due to greediness might select morpheme B before morpheme A because B had a higher weight while A was ultimately correct. To select A instead of B, the morpheme candidates would have to be reordered which, without an efficient way to propose these order, worsens the time complexity from $O(n^c)$ to $O(n!)$. Related algorithms use iterative approaches (IBM models and Morfessor) or search [Brighton, 2003a] to avoid the local minima that trap greedy approaches. Future work could incorporate such methods to improve upon the performance of CSAR for morpheme induction.

Bias towards smaller forms and meanings In light of the greedy approach, a further limitation of CSAR is its bias towards smaller forms and meanings due to the fact that mutual information is intrinsic bias towards frequently appearing forms and meanings (n.b.: smaller forms necessarily appear as or more often than larger, composite forms. While this bias is beneficial where emergent languages are genuinely decomposable into smaller units, in languages which do not decompose effectively, CSAR induces low quality inventories (i.e., having low morpheme bijectivities) with small forms and meanings rather than inducing more bijective inventories with holistic meanings (i.e., larger forms and meanings). Some heuristics balancing mutual information and bijectivity could address this while maintaining greediness, but ultimately non-greedy approaches might be necessary.

Limited emergent language data The other limitation of this paper relates to the type and breadth of emergent language data. In terms of type, since we do not have ground truth morpheme inventories for emergent language, we cannot directly evaluate CSAR's performance on the target domain (hence the validation with procedurally generated and human languages). In terms of breadth, without a larger and more representative sample of more systematically generated data we are unable to make definitive claims about the patterns and trends of morpheme inventories in emergent languages.

Chapter 7

Detecting Structure Among Morphemes [proposed]

7.1 Introduction

Our goal in this chapter is to introduce an algorithm which discover structural patterns among morphemes in emergent language corpora which been segmented into morphemes. This a step in the larger project of developing methods to identify the syntactic structure of emergent languages ultimately so as to compare emergent languages to human languages. This chapter makes minimal claims as to how similar the structural patterns it studies are to syntax in a fuller linguistic sense. We take this minimalist approach as the structural characteristics of emergent languages are largely unknown, so we make as few assumptions as possible as to how emergent languages are structured.[1]

Related Work The syntactic structure of emergent language has been studied before from a few different angles. Chaabouni et al. [2019c] look at the ordering of words in emergent language to see if they resemble human language. The work proposed in this chapter differs primarily in scope where instead of identifying one type of structure in one particular emergent language, this presented algorithm is intended to generalize across both structural patterns and emergent language environments.

Grammar induction both inside and outside of emergent communication is a closely related area. For example, van der Wal et al. [2020] use unsupervised grammar induction algorithms to analyze the structure of emergent languages. Similarly, Ueda et al. [2022] use supervised categorial grammar induction to study the compositionality of emergent languages. Apart from emergent language, grammar induction has been studied directly on human languages [Kim et al., 2019b,a] as well as on artificial languages [Kirby, 2002, Brighton, 2003b]. The work in this chapter differs from these more typical grammar induction tasks because it makes fewer assumptions about the existence of structure at all in emergent languages. Rather than

[1] Author note: I am on the fence as to whether to incorporate semantics into this chapter on structure. The primary way I would imagine doing this is by taking the structural feature functions and contextualizing in terms of certain meanings; that is, we ask how often a structural feature is present *and* used to convey a certain meaning (which is a function of its parts). I am leaning against this extension right now because I think this chapter has enough moving parts as it is, and I think it would be good to walk before running.

assuming that a structure exists and trying to uncover it, the algorithm attempts to detect if there is any structure at all and, only if so, determine what it is.

7.2 Algorithm

The algorithm takes the following as input, (1) a corpus of utterances where each component is a morpheme (see Chapter 6), (2) a mapping from morphemes to morpheme classes, (3) a set of structural feature functions. The output of the algorithm is a list of structural features which apply to certain classes of morphemes across the corpus, giving a primitive notion of the "syntax" of the emergent language described in the corpus.

Morphemes

It is assumed that each utterance in the corpus is a sequence of morphemes, notated as follows

$$U = (u_1, u_2, \ldots, u_{|U|}) \tag{7.1}$$

$$\text{where} \quad u_i \in \mathcal{M}, \tag{7.2}$$

where $\mathcal{M}$ is the set of morphemes. Additionally, each morpheme can belong to one or more "classes" corresponding the nature of the morpheme itself. Most commonly, classes would be derived from the semantic features of the morpheme; for example, in a colored shape naming game, one morpheme class would be *color* and another *shape* (assuming the language fully compositional). Classes could also be based on the form of the morpheme like *single-token* or *double-token*. Finally, the morpheme extraction step might yield tokens which do not correspond to any meaning (that the algorithm could find); these morphemes will map to an "unclassified" catch-all class. These classes will later be used to determine if certain patterns occur only at the level of individual morphemes or at the level of certain classes of morphemes. This morpheme classification mapping is written as

$$C : \mathcal{M} \to \mathcal{P}(\mathcal{C}) \setminus \varnothing, \tag{7.3}$$

where $\mathcal{P}(\cdot)$ is the power set operator and $\mathcal{C}$ is the set of all classes so the co-domain of the function is all non-empty subsets of $\mathcal{C}$.

Structural feature functions

The most important input to the algorithm is the set of *structural feature functions* which detect the presence of certain structural features in a given utterance. A feature function is specifically a boolean-valued logical formula which takes as arguments an utterance and a tuple of morpheme classes depending on the arity of the function. Thus, we write

$$F : U \times \mathcal{C}^a \to \{0, 1\}, \tag{7.4}$$

where a is the arity of the function.

Using this formalism, we can define a structural feature function which detects whether a morpheme class occurs at the beginning of an utterance as

$$\textsc{Begin}(U, c_1) \equiv c_1 \in C(u_1). \tag{7.5}$$

We can express the same function more succinctly with some abuse of notation:

$$\textsc{Begin}(c_1) \equiv \textsc{Begin}^1 \equiv c_1(u_1), \tag{7.6}$$

where $\textsc{Begin}^n$ would mean that the function has an arity of n with the convention that the classes are named c_1, c_2, and so on until c_n and $c_1(u_1)$ is true iff u_1 belongs to class c_1. Since all feature functions take a single utterance as input, the U argument is implicit. For some feature functions, it is handy to generalize them by parameterizing the function itself. For example, if we want a function detecting the absolute position of a morpheme class, we would write

$$\textsc{AbsPos}(i)^1 \equiv (\textsc{AbsPos}(i))(c_1) \equiv c_1(u_i) \tag{7.7}$$

Note that the higher-order parameters of the function precede the superscript to distinguish The position of arguments before the superscript i

Finally, in some cases, it may be the case that structural feature of interest depends on a particular morpheme and not its class generally. In such cases, the feature function could take a morpheme m instead of a morpheme class c as an argument, and occurrences of $c(u)$ in the formula could be replaced with $m = u$ (where u is morpheme from the utterance).

To illustrate the use of this formalism of structural feature functions we will define a handful of common structural features from the syntax of human language. These functions will also be used in the experiments presented in Section 7.3.

Absolute position Defined above.

Relative position We define immediate precession of classes as

$$\textsc{Precede}^2 \equiv \exists i \; c_1(u_i) \wedge c_2(u_{i+1}). \tag{7.8}$$

More flexibly, if one classes occurs earlier in the sequence than another class (possibly non-immediately), we write

$$\textsc{Before}^2 \equiv \exists i, j \; i < j \wedge c_1(u_i) \wedge c_2(u_j). \tag{7.9}$$

Naturally, reversing the order arguments yields $\textsc{Succeed}^2$ and $\textsc{After}^2$. Finally, we could generalize relative positioning to any number of morpheme classes with

$$\textsc{Order}^n \equiv \exists i \; \bigwedge_{j=1}^{n} c_j(u_{i+j-1}), \tag{7.10}$$

where n is the number of morpheme classes in the specified ordering.

Occurrence More generally, we define occurrence of a morpheme class at any place in sequence with

$$\textsc{Occur}^1 \equiv \exists i \; c_1(u_i). \tag{7.11}$$

For the co-occurrence of two morpheme classes, we write

$$\textsc{CoOccur}^2 \equiv \exists i, j \; i \neq j \wedge c_1(u_i) \wedge c_2(u_j). \tag{7.12}$$

Note that we must exclude the possibility of the individual occurrences being the same morpheme in the utterance. The definition of the co-occurrence of an arbitrary number of morpheme classes is an exercise left to the reader.

Linking Moving in the direction of more sophisticated linguistic concepts, we introduce the $\textsc{Link}^3$ function which defines a notion of the presence of two morphemes requiring the presence of a third morpheme:

$$\textsc{Link}^3 \equiv \textsc{CoOccur}(c_1, c_2) \rightarrow \textsc{Occur}(c_3) \tag{7.13}$$

$$\textsc{Link}^3 \equiv \exists i, j, k \; \text{distinct}(i, j, k) \wedge (c_1(u_i) \wedge c_2(u_j) \rightarrow c_3(u_k)), \tag{7.14}$$

where distinct$(\cdot)$ denotes that no two argument are equal. Some examples of syntactic rules which are approximated by this definition include conjunctions in noun phrases in English (e.g., "dog" and "cat" occurring in a noun phrase requires something like "and" or "or" to join them) or verb roots in Latin requiring a finite ending to agree with a noun as its subject (e.g., the verb root "sta-" ("to stand") requires the ending "-t" to agree with "canis" (dog) in "canis stat" (the dog stands)).

Identifying common structures

While the above structural feature functions identify when a particular structure appears in a given utterance, the goal of the algorithm is to identify structures that characterize the emergent language as a whole (or at least the corpora). Thus, in this section, we define the part of the algorithm that is responsible for aggregating the results of individual utterances so as to determine what patterns are significant across the entire corpus.

To start, we can see that when the above feature functions are run across a corpus of utterances, the result is essentially the numerator of a probability measure, that is, the total number of times the structure occurs. The second component is, of course, the denominator, which denotes the number of times an event could have occurred, that is, the number of times it occurs plus the number of times it could have occurred but did not. For each feature function, then, we need to describe some counterfactual notion of occurrence (i.e., "could have occurred but did not").

We provide a list of these denominator functions for each structural feature function in Table 7.1. For the most part, the denominator functions are straightforward: for $\textsc{CoOccur}^2$, we compare this to how many times either one of the arguments occurs at all (e.g., adjectives generally co-occur with nouns in English) and for functions like $\textsc{Precede}^2$, we compare it to how many times the argument co-occur in any order (e.g., if an adjective modifies a noun

Feature	Denominator
$\textsc{Occur}^1$	$\textsc{True}^0$
$\textsc{CoOccur}^2$	$\textsc{Occur}(c_1) \vee \textsc{Occur}(c_2)$
$\textsc{Begin}^1$	$\textsc{True}^0$
$\textsc{Begin}^1$	$\textsc{Occur}(c_1)$
$\textsc{AbsPos}(i)^1$	$\textsc{Occur}(c_1)$
$\textsc{Precede}^2$	$\textsc{CoOccur}(c_1, c_2)$
$\textsc{Before}^2$	$\textsc{CoOccur}(c_1, c_2)$
$\textsc{Order}^n$	$\textsc{CoOccur}(c_1, c_2, \ldots, c_n)$
$\textsc{Link}^3$	$\textsc{CoOccur}(c_1, c_2)$

Table 7.1: Table of denominator functions for each structural feature function.

in English, it precedes it). Less intuitive cases include the following: $\textsc{Occur}^1$ is paired with the trivial function $\textsc{True}^0$ which effectively divides the numerator by the total number of utterances. $\textsc{Begin}^1$ can have two different interpretations based on the denominator used: this morpheme class begins every sentence (denominator of $\textsc{True}^0$ versus this morpheme is always at the beginning when it is present (denominator of $\textsc{Occur}^1$). This distinction could potentially be made with other feature functions. Finally, $\textsc{Link}^3$ derives its denominator from the antecedent of the implication to ignore trivial satisfaction of the implication (i.e., False $\rightarrow q \Leftrightarrow$ True).

Thus, we can define the probability of a structural feature function f holding for classes $c_1, \ldots, c_n$ as

$$p_f(c_1, \ldots, c_n) \equiv \frac{\sum_{U \in \mathcal{U}} f(U, c_1, \ldots, c_n)}{\sum_{U \in \mathcal{U}} d(U, c_1, \ldots, c_n)}, \tag{7.15}$$

where $\mathcal{U}$ is a collection of utterances (i.e., the corpus) and d is the denominator function corresponding to f. Finally, if a rule holds with a probability above a threshold t, we consider to hold for the corpus generally. We, then, define the set of rules which characterize the corpus to be

$$\{f(c_1, \ldots, c_n) \mid p_f(c_1, \ldots, c_n) \wedge f \in \mathcal{F} \wedge c_1, \ldots, c_n \in \mathcal{C}\}, \tag{7.16}$$

where $\mathcal{F}$ is the set of all structural feature functions and $\mathcal{C}$ is the set of all morpheme classes.

Recursion

The above process could potentially be applied recursively where instances of morpheme patterns are replaced with with with a corresponding "morpheme" token (i.e., a non-terminal symbol in formal grammar terms). This would yield the potential to find higher-level patterns, if they exist. Nevertheless, recursive applications could also suffer from compounding errors in the applications the structure detection algorithm. For example, if the threshold is set too high for what counts as a structural pattern, patterns will be missed and higher-level rules will not be discovered; conversely, too low a threshold will yield patterns that are not

genuinely descriptive while giving the illusion of documenting the structure of emergent languages.

7.3 Experiments

Data

The experiments proposed for this chapter largely mirror those in Chapter 6. Namely, the morpheme structure algorithm will be run across a handful of synthetic languages as well as a handful of real emergent corpora in order to illustrate results on real data. The synthetic data will be generated via rule-based methods such as probabilistic context-free grammars (PCFGs) in order to establish an *a prior* set of patterns to compare the results against. The emergent language corpora will be derived from the results of algorithm from Chapter 6 which are, in turn, derived from ELCC Plus (Chapter 3).

Analysis

The analysis of the results on the synthetic data will focus primarily on whether or not the algorithm's results match the *a prior* expectations of the given how the data was generated. That is, we expect the algorithm to recover the rules present in the grammar (or at least the simpler ones) while not identifying spurious rules. Furthermore, these analyses will also look at to what degree the algorithm is sensitive to noise; for example, if random, unclassified morphemes are added to the grammar's outputs, we will determine to what degree the algorithm's outputs are degraded.

Regarding the analyses on emergent language corpora: whereas the morpheme detection involves emergent communication system-specific elements (i.e., defining what a "meaning" is and to what classes it belongs), the morpheme structure detection is agnostic to environment as it deals only with morphemes and morpheme classes in the general case. Thus, the analysis in this chapter will focus less on the application of the algorithm to different environments and more on the particular patterns seen in the results. The primary questions to be addressed on this front are (1) whether structural patterns are detected at all, (2) whether these patterns match any *a priori* expectations, and (3) whether and how these patterns vary between different systems.

Chapter 8

Conclusion

[:todo]

Bibliography

Mubashara Akhtar, Omar Benjelloun, Costanza Conforti, Pieter Gijsbers, Joan Giner-Miguelez, Nitisha Jain, Michael Kuchnik, Quentin Lhoest, Pierre Marcenac, Manil Maskey, Peter Mattson, Luis Oala, Pierre Ruyssen, Rajat Shinde, Elena Simperl, Goeffry Thomas, Slava Tykhonov, Joaquin Vanschoren, Jos van der Velde, Steffen Vogler, and Carole-Jean Wu. Croissant: A metadata format for ml-ready datasets. DEEM '24, page 1–6, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706110. doi: 10.1145/3650203.3663326. URL https://doi.org/10.1145/3650203.3663326.

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

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.421. URL https://aclanthology.org/2020.acl-main.421.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv*, 1909.07528, 2020. URL https://arxiv.org/abs/1909.07528.

Khuyagbaatar Batsuren, Gábor Bella, Aryaman Arora, Viktor Martinovic, Kyle Gorman, Zdeněk Žabokrtský, Amarsanaa Ganbold, Šárka Dohnalová, Magda Ševčíková, Kateřina Pelegrinová, Fausto Giunchiglia, Ryan Cotterell, and Ekaterina Vylomova. The SIGMORPHON 2022 shared task on morpheme segmentation. In Garrett Nicolai and Eleanor Chodroff, editors, *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 103–116, Seattle, Washington, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.sigmorphon-1.11. URL https://aclanthology.org/2022.sigmorphon-1.11/.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.703. URL https://www.aclweb.org/anthology/2020.emnlp-main.703.

Frederic Blum, Carlos Barrientos, Adriano Ingunza, Damián E Blasi, and Roberto Zariquiey. Grammars across time analyzed (GATA): a dataset of 52 languages. *Scientific Data*, 10(1): 835, 2023.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

Brendon Boldt and David Mortensen. Mathematically modeling the lexicon entropy of emergent language. *arXiv*, 2211.15783, 2023. URL https://arxiv.org/abs/2211.15783.

Brendon Boldt and David Mortensen. ELCC: the Emergent Language Corpus Collection. *arXiv*, 2407.04158, 2024a. URL https://arxiv.org/abs/2407.04158.

Brendon Boldt and David Mortensen. XferBench: a data-driven benchmark for emergent language. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *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, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.82. URL https://aclanthology.org/2024.naacl-long.82.

Brendon Boldt and David R Mortensen. A review of the applications of deep learning-based emergent communication. *Transactions on Machine Learning Research*, 2024c. ISSN 2835-8856. URL https://openreview.net/forum?id=jesKcQxQ7j.

Diane Bouchacourt and Marco Baroni. How agents see things: On visual representations in an emergent language game. *arXiv*, arXiv:1808.10696, 2018.

Nicolo' Brandizzi, Davide Grossi, and Luca Iocchi. RLupus: Cooperation through the emergent communication in The Werewolf social deduction game. *Intelligenza Artificiale*, 15(2):55–70, 2022. URL https://content.iospress.com/articles/intelligenza-artificiale/ia210081.

Henry Brighton. *Simplicity as a driving force in linguistic evolution*. PhD thesis, University of Edinburgh, Edinburgh, UK, 2003a. URL https://era.ed.ac.uk/handle/1842/23810.

Henry Brighton. *Simplicity as a driving force in linguistic evolution*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, UK, 2003b. URL http://hdl.handle.net/1842/23810.

Henry Brighton and Simon Kirby. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial Life*, 12:229–242, 2006a.

Henry Brighton and Simon Kirby. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial Life*, 12(2):229–242, 2006b. doi: 10.1162/artl.2006.12.2.229.

Henry Brighton, Kenny Smith, and Simon Kirby. Language as an evolutionary system. *Physics of Life Reviews*, 2(3):177–226, 2005. ISSN 1571-0645. doi: https://doi.org/10.1016/j.plrev.2005.06.001. URL https://www.sciencedirect.com/science/article/pii/S1571064505000229.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL https://aclanthology.org/J93-2003/.

Kalesha Bullard, Franziska Meier, Douwe Kiela, Joelle Pineau, and Jakob N. Foerster. Exploring zero-shot emergent communication in embodied multi-agent populations. *ArXiv*, abs/2010.15896, 2020. URL https://api.semanticscholar.org/CorpusID:226222037.

Kalesha Bullard, Douwe Kiela, Franziska Meier, Joelle Pineau, and Jakob Foerster. Quasi-equivalence discovery for zero-shot emergent communication. *arXiv*, arXiv:2103.08067, 2021.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. *arXiv*, 2012.07805, 2021. URL https://arxiv.org/abs/2012.07805.

Boaz Carmeli, Ron Meir, and Yonatan Belinkov. Emergent quantized communication. *arXiv*, arXiv:2211.02412, 2022.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. Anti-efficient encoding in emergent communication. *arXiv*, arXiv:1905.12561, 2019a.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. *Anti-efficient encoding in emergent communication*. Curran Associates Inc., Red Hook, NY, USA, 2019b.

Rahma Chaabouni, Eugene Kharitonov, Alessandro Lazaric, Emmanuel Dupoux, and Marco Baroni. Word-order biases in deep-agent emergent communication. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5166–5175, Florence, Italy, July 2019c. Association for Computational Linguistics. doi: 10.18653/v1/P19-1509. URL https://aclanthology.org/P19-1509.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. *arXiv*, arXiv:2004.09124, 2020.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. Communicating artificial neural networks develop efficient color-naming systems. *Proceedings of the National Academy of Sciences*, 118(12):e2016569118, 2021. doi: 10.1073/pnas.2016569118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2016569118.

Rahma Chaabouni, Florian Strub, Florent Altché, Eugene Tarassov, Corentin Tallec, Elnaz Davoodi, Kory Wallace Mathewson, Olivier Tieleman, Angeliki Lazaridou, and Bilal Piot. Emergent communication at scale. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AUGBfDIV9rL.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Aritra Chowdhury, Alberto Santamaria-Pang, James R. Kubricht, Jianwei Qiu, and Peter Tu. Symbolic semantic segmentation and interpretation of covid-19 lung infections in chest ct volumes based on emergent languages. *arXiv*, arXiv:2008.09866, 2020a.

Aritra Chowdhury, Alberto Santamaria-Pang, James R. Kubricht, and Peter Tu. Emergent symbolic language based deep medical image classification. *arXiv*, arXiv:2008.09860, 2020b.

Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics, July 2002. doi: 10.3115/1118647.1118650. URL https://aclanthology.org/W02-0603/.

Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. Co-evolution of language and agents in referential games. *arXiv*, arXiv:2001.03361, 2020.

Kevin Denamganaï and James Alfred Walker. On (emergent) systematic generalisation and compositionality in visual referential games with straight-through gumbel-softmax estimator. *arXiv*, arXiv:2012.10776, 2020.

Roberto Dessì, Diane Bouchacourt, Davide Crepaldi, and Marco Baroni. Focus on what's informative and ignore what's not: Communication strategies in a referential game. *arXiv*, arXiv:1911.01892, 2019.

Roberto Dessì, Eugene Kharitonov, and Marco Baroni. Interpretable agent communication from scratch (with a generic visual processor emerging on the side). *arXiv*, arXiv:2106.04258, 2021.

C.m. Downey, Xuhui Zhou, Zeyu Liu, and Shane Steinert-Threlkeld. Learning to translate by learning to communicate. In Duygu Ataman, editor, *Proceedings of the 3rd Workshop on Multi-lingual Representation Learning (MRL)*, pages 218–238, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.mrl-1.17.

Matthew S. Dryer and Martin Haspelmath, editors. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013. URL https://wals.info/.

Kevin Eloff, Arnu Pretorius, Okko Johannes Räsänen, Herman Arnold Engelbrecht, and Herman Kamper. Towards learning to speak and hear through multi-agent communication over a continuous acoustic channel. *ArXiv*, abs/2111.02827, 2021. URL https://api.semanticscholar.org/CorpusID:242757488.

Kevin Eloff, Okko Räsänen, Herman A. Engelbrecht, Arnu Pretorius, and Herman Kamper. Towards learning to speak and hear through multi-agent communication over a continuous acoustic channel. *arXiv*, 2111.02827, 2023.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2145–2153, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Philip Gage. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38, 1994a. URL https://api.semanticscholar.org/CorpusID:59804030.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994b. ISSN 0898-9788.

Lukas Galke, Yoav Ram, and Limor Raviv. Emergent communication for understanding human language evolution: What's missing? *ArXiv*, abs/2204.10590, 2022.

Edward Gibson, Richard Futrell, Julian Jara-Ettinger, Kyle Mahowald, Leon Bergen, Sivalogeswaran Ratnasingam, Mitchell Gibson, Steven T. Piantadosi, and Bevil R. Conway. Color naming across languages reflects color use. *Proceedings of the National Academy of Sciences*, 114(40):10785–10790, 2017. doi: 10.1073/pnas.1619666114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1619666114.

Shangmin Guo, Yi Ren, Serhii Havrylov, Stella Frank, Ivan Titov, and Kenny Smith. The emergence of compositional languages for numeric concepts through iterated learning in neural agents. *arXiv*, arXiv:1910.05291, 2019.

Shangmin Guo, Yi Ren, Agnieszka Słowik, and Kory Mathewson. Inductive bias and language expressivity in emergent communication. *arXiv*, arXiv:2012.02875, 2020.

Yuxuan Guo, Yifan Hao, Rui Zhang, Enshuai Zhou, Zidong Du, Xishan Zhang, Xinkai Song, Yuanbo Wen, Yongwei Zhao, Xuehai Zhou, Jiaming Guo, Qi Yi, Shaohui Peng, Di Huang, Ruizhi Chen, Qi Guo, and Yunji Chen. Emergent communication for rules reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=gx20B4ItIw.

Abhinav Gupta, Cinjon Resnick, Jakob Foerster, Andrew Dai, and Kyunghyun Cho. Compositionality and capacity in emergent languages. In Spandana Gella, Johannes Welbl, Marek Rei, Fabio Petroni, Patrick Lewis, Emma Strubell, Minjoon Seo, and Hannaneh Hajishirzi, editors, *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 34–38, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.repl4nlp-1.5. URL https://aclanthology.org/2020.repl4nlp-1.5.

Zellig S. Harris. From phoneme to morpheme. *Language*, 31(2):190–222, 1955. ISSN 00978507, 15350665. URL http://www.jstor.org/stable/411036.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *arXiv*, arXiv:1705.11192, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z. Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3040–3049. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/jaques19a.html.

Yipeng Kang, Tonghan Wang, and Gerard de Melo. Incorporating pragmatic reasoning communication into emergent language, 2020. URL https://arxiv.org/abs/2006.04109.

Bence Keresztury and Elia Bruni. Compositional properties of emergent languages in deep learning. *arXiv*, arXiv:2001.08618, 2020.

Eugene Kharitonov and Marco Baroni. Emergent language generalization and acquisition speed are not tied to compositionality. *arXiv*, arXiv:2004.03420, 2020a.

Eugene Kharitonov and Marco Baroni. Emergent language generalization and acquisition speed are not tied to compositionality. *arXiv*, 2004.03420, 2020b. URL https://arxiv.org/abs/2004.03420.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. EGG: a toolkit for research on emergence of lanGuage in games. In Sebastian Padó and Ruihong Huang, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 55–60, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-3010. URL https://aclanthology.org/D19-3010.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. Entropy minimization in emergent languages. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5220–5230. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kharitonov20a.html.

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

Bohdan Khomtchouk and Shyam Sudhakaran. Modeling natural language emergence with integral transform theory and reinforcement learning. *arXiv*, arXiv:1812.01431, 2018.

Yoon Kim, Chris Dyer, and Alexander Rush. Compound probabilistic context-free grammars for grammar induction. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1228. URL https://aclanthology.org/P19-1228/.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. Unsupervised recurrent neural network grammars. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota, June 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1114. URL https://aclanthology.org/N19-1114/.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Simon Kirby. *Learning, bottlenecks and the evolution of recursive syntax*, page 173–204. Cambridge University Press, 2002.

Tomasz Korbak, Julian Zubek, and Joanna Rączaszek-Leonardi. Measuring non-trivial compositionality in emergent communication, 2020. URL https://arxiv.org/abs/2010.15058.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1321. URL https://aclanthology.org/D17-1321/.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://aclanthology.org/P18-1007/.

Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, IL, 1962.

Alexander Kuhnle and Ann Copestake. Shapeworld - a new test methodology for multimodal language understanding. *arXiv*, 1704.04517, 2017a. URL https://arxiv.org/abs/1704.04517.

Alexander Kuhnle and Ann A. Copestake. Shapeworld - a new test methodology for multimodal language understanding. *ArXiv*, abs/1704.04517, 2017b. URL https://api.semanticscholar.org/CorpusID:16515835.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. Morpho challenge 2005-2010: Evaluations and results. In Jeffrey Heinz, Lynne Cahill, and Richard Wicentowski, editors, *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/W10-2211/.

Travis LaCroix. Biology and compositionality: Empirical considerations for emergent-communication protocols. *CoRR*, abs/1911.11668, 2019. URL http://arxiv.org/abs/1911.11668.

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: a llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340052. doi: 10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162.

Nur Geffen Lan, Emmanuel Chemla, and Shane Steinert-Threlkeld. On the spontaneous emergence of discrete and compositional signals. *arXiv*, arXiv:2005.00110, 2020.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. *arXiv*, 2006.02419, 2020. URL https://arxiv.org/abs/2006.02419.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv*, 1612.07182, 2016.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *ArXiv*, abs/1804.03984, 2018a.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv*, 1804.03984, 2018b. URL https://arxiv.org/abs/1804.03984.

David Kellogg Lewis. *Convention: A Philosophical Study*. Wiley-Blackwell, Cambridge, MA, USA, 1969.

Fushan Li and Michael Bowling. Ease-of-teaching and language structure from emergent communication. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Yaoyiran Li, Edoardo Maria Ponti, Ivan Vulić, and Anna Korhonen. Emergent communication pretraining for few-shot machine translation. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4716–4731, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.416. URL https://aclanthology.org/2020.coling-main.416.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. *arXiv*, 1405.0312, 2015. URL https://arxiv.org/abs/1405.0312.

Olaf Lipinski, Adam Sobey, Federico Cerutti, and Timothy J. Norman. Speaking your language: Spatial relationships in interpretable emergent communication. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=vIP8IWmZlN.

Ryan Lowe, Abhinav Gupta, Jakob Foerster, Douwe Kiela, and Joelle Pineau. On the interaction between supervision and self-play in emergent communication. *arXiv*, arXiv:2002.01093, 2020.

Diana Rodríguez Luna, Edoardo Maria Ponti, Dieuwke Hupkes, and Elia Bruni. Internal and external pressures on language emergence: least effort, object constancy and frequency. *arXiv*, arXiv:2004.03868, 2020.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Matéo Mahaut, Francesca Franzon, Roberto Dessì, and Marco Baroni. Referential communication in heterogeneous communities of pre-trained visual deep networks. *arXiv*, arXiv:2302.08913, 2023.

Benoit Mandelbrot et al. An informational theory of the statistical structure of language. *Communication theory*, 84:486–502, 1953.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19:313–330, 1993. URL https://api.semanticscholar.org/CorpusID:252796.

Daniela Mihai and Jonathon Hare. Avoiding hashing and encouraging visual semantics in referential emergent language games. *arXiv*, arXiv:1911.05546, 2019.

Daniela Mihai and Jonathon Hare. The emergence of visual semantics through communication games. *arXiv*, arXiv:2101.10253, 2021a.

Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. *arXiv*, 2106.02067, 2021b.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Clément Moulin-Frier and Pierre-Yves Oudeyer. Multi-agent reinforcement learning as a computational tool for language evolution research: Historical context and future challenges. *ArXiv*, abs/2002.08878, 2020.

Jesse Mu and Noah Goodman. Emergent communication of generalizations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17994–18007. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9597353e41e6957b5e7aa79214fcb256-Paper.pdf.

Jesse Mu and Noah Goodman. Emergent communication of generalizations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17994–18007. Curran Associates, Inc., 2021b. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9597353e41e6957b5e7aa79214fcb256-Paper.pdf.

Yao Mu, Shunyu Yao, Mingyu Ding, Ping Luo, and Chuang Gan. Ec2: Emergent communication for embodied control. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6704–6714, 2023. doi: 10.1109/CVPR52729.2023.00648.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational*

Linguistics, 3:157–167, 2015. doi: 10.1162/tacl_a_00130. URL https://aclanthology.org/Q15-1012/.

Xenia Ohmer, Michael Marino, Michael Franke, and Peter König. Mutual influence between language and perception in multi-agent communication games. *arXiv*, arXiv:2112.14518, 2021. doi: 10.1371/journal.pcbi.1010658.

Xenia Ohmer, Marko Duda, and Elia Bruni. Emergence of hierarchical reference systems in multi-agent communication. *arXiv*, arXiv:2203.13176, 2022.

Isabel Papadimitriou and Dan Jurafsky. Learning music helps you read: Using transfer to study linguistic structure in language models. In *Conference on Empirical Methods in Natural Language Processing*, 2020. URL https://api.semanticscholar.org/CorpusID:221891676.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Hugh Perkins. Neural networks can understand compositional functions that humans do not, in the context of emergent communication. *arXiv*, arXiv:2103.04180, 2021a.

Hugh Perkins. Texrel: a green family of datasets for emergent communications on relations. *arXiv preprint arXiv:2105.12804*, 2021b.

Hugh Perkins. Icy: A benchmark for measuring compositional inductive bias of emergent communication models, 2022. URL https://openreview.net/forum?id=S352vriz3G.

S.T. Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. *Psychon Bull Rev*, 21:1112—1130, 2014. URL https://doi.org/10.3758/s13423-014-0585-6.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL https://aclanthology.org/W15-3049.

Eva Portelance, Michael C. Frank, Dan Jurafsky, Alessandro Sordoni, and Romain Laroche. The emergence of the shape bias results from communicative efficiency. *arXiv*, arXiv:2109.06232, 2021.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B. Cohen, and Simon Kirby. Compositional languages emerge in a neural iterated learning model. *arXiv*, arXiv:2002.01365, 2020.

Mathieu Rita, Rahma Chaabouni, and Emmanuel Dupoux. "lazimpa": Lazy and impatient neural agents learn to communicate efficiently. *arXiv*, arXiv:2010.01878, 2020.

Mathieu Rita, Florian Strub, Jean-Bastien Grill, Olivier Pietquin, and Emmanuel Dupoux. On the role of population heterogeneity in emergent communication. *arXiv*, arXiv:2204.12982, 2022a.

Mathieu Rita, Corentin Tallec, Paul Michel, Jean-Bastien Grill, Olivier Pietquin, Emmanuel Dupoux, and Florian Strub. Emergent communication: Generalization and overfitting in lewis games. *arXiv*, arXiv:2209.15342, 2022b.

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

M.P. Schützenberger. On context-free languages and push-down automata. *Information and Control*, 6(3):246–264, 1963. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(63)90306-1. URL https://www.sciencedirect.com/science/article/pii/S0019995863903061.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162/.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1238. URL https://aclanthology.org/P18-1238.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv*, 1712.01815, 2017. URL https://arxiv.org/abs/1712.01815.

Shane Steinert-Threlkeld. Paying attention to function words. *arXiv*, arXiv:1909.11060, 2019.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. LSTM networks can perform dynamic counting. In Jason Eisner, Matthias Gallé, Jeffrey Heinz, Ariadna Quattoni, and Guillaume Rabusseau, editors, *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3905. URL https://aclanthology.org/W19-3905.

Agnieszka Słowik, Abhinav Gupta, William L. Hamilton, Mateja Jamnik, Sean B. Holden, and Christopher Pal. Structural inductive biases in emergent communication. *arXiv*, arXiv:2002.01335, 2020.

Kumiko Tanaka-Ishii. *Articulation of Elements*, pages 115–124. Springer International Publishing, Cham, 2021. ISBN 978-3-030-59377-3. doi: 10.1007/978-3-030-59377-3_11. URL https://doi.org/10.1007/978-3-030-59377-3_11.

J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. PettingZoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

Mycal Tucker, Huao Li, Siddharth Agrawal, Dana Hughes, Katia Sycara, Michael Lewis, and Julie A Shah. Emergent discrete communication in semantic spaces. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 10574–10586. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/5812f92450ccaf17275500841c70924a-Paper.pdf.

Ryo Ueda, Taiga Ishii, Koki Washio, and Yusuke Miyao. Categorial grammar induction as a compositionality measure for emergent languages in signaling games. In *Emergent Communication Workshop at ICLR 2022*, 2022. URL https://openreview.net/forum?id=Sbgb7b0Q-5.

Ryo Ueda, Taiga Ishii, and Yusuke Miyao. On the word boundaries of emergent languages based on harris's articulation scheme. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b4t9_XASt6G.

Thomas A. Unger and Elia Bruni. Generalizing emergent communication. *arXiv: Artificial Intelligence*, 2020. URL https://arxiv.org/abs/2001.01772.

Oskar van der Wal, Silvan de Boer, Elia Bruni, and Dieuwke Hupkes. The grammar of emergent languages. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3339–3359, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.270. URL https://aclanthology.org/2020.emnlp-main.270.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Python implementation and extensions for Morfessor baseline. Technical Report ISBN 978-952-60-5501-5, Aalto University, Helsinki, Finland, 2013.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018. URL https://api.semanticscholar.org/CorpusID:5034059.

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

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Ethical and social risks of harm from language models. *arXiv*, 2112.04359, 2021. URL https://arxiv.org/abs/2112.04359.

Gregory M. Werner and Michael G. Dyer. Evolution of communication in artificial organisms. *Artificial Life II*, pages 659 – 687, 1991.

Wikimedia Foundation. Wikimedia downloads. URL https://dumps.wikimedia.org.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019. URL https://api.semanticscholar.org/CorpusID:208117506.

Shunyu Yao, Mo Yu, Yang Zhang, Karthik R Narasimhan, Joshua B. Tenenbaum, and Chuang Gan. Linking emergent and natural languages via corpus transfer. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=49A1Y6tRhaq.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. Minari, September 2024. URL https://doi.org/10.5281/zenodo.13767625.

Noga Zaslavsky, Charles Kemp, Naftali Tishby, and Terry Regier. Color naming reflects both perceptual structure and communicative need. *Topics in Cognitive Science*, 11(1):207–219, 2019. doi: https://doi.org/10.1111/tops.12395. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12395.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

George Kingsley Zipf. *Human behavior and the principle of least effort.* Addison-Wesley Press, Oxford, England, 1949.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. URL https://aclanthology.org/D16-1163.

Łukasz Kuciński, Tomasz Korbak, Paweł Kołodziej, and Piotr Miłoś. Catalytic role of noise and necessity of inductive biases in the emergence of compositional communication. *arXiv*, arXiv:2111.06464, 2021.

Appendix A

ELCC

A.1 ECS-Level Metadata Specification

source The URL for the repository implementing the ECS.

upstream_source The URL of the original repo if **source** is a fork.

paper The URL of the paper documenting the ECS (if any).

game_type The high level category of the game implemented in the ECS; currently one of *signalling*, *conversation*, or *navigation*.

game_subtype A finer-grained categorization of the game, if applicable.

observation_type The type of observation that the agents make; currently either *vector* or *image* (i.e., an image embedding).

observation_continuous Whether or not the observation is continuous as opposed to discrete (e.g., image embeddings versus concatenated one-hot vectors).

data_source Whether the data being communicated about is from a natural source (e.g., pictures), is synthetic, or comes from another source (e.g., in a social deduction game).

variants A dictionary where each entry corresponds to one of the variants of the particular ECS. Each entry in the dictionary contains any relevant hyperparameters that distinguish it from the other variants.

seeding_available Whether or not the ECS implements seeding the random elements of the system.

multi_step Whether or not the ECS has multiple steps per episode.

symmetric_agents Whether or not agents both send and receive messages.

multi_utterance Whether or not multiple utterances are included per line in the dataset.

more_than_2_agents Whether or not the ECS has a population of >2 agents.

```yaml
origin:
  upstream_source:
    https://github.com/google-deepmind/emergent_communication...
  paper: https://openreview.net/forum?id=AUGBfDIV9rL
system:
  game_type: signalling
  data_source: natural
  game_subtype: discrimination
  observation_type: image
  observation_continuous: true
  seeding_available: true
  multi_step: false
  more_than_2_agents: true
  multi_utterance: false
  symmetric_agents: false
  variants:
    imagenet-1x10:
      n_receivers: 10
      n_senders: 1
    imagenet-10x10:
      n_receivers: 10
      n_senders: 10
    imagenet-5x5:
      n_receivers: 5
      n_senders: 5
    imagenet-1x1:
      n_receivers: 1
      n_senders: 1
    imagenet-10x1:
      n_receivers: 1
      n_senders: 10
```

Figure A.1: Example of an ECS metadata file in the YAML format.

A.2 ECS-Level Metadata Example

See Figure A.1.

A.3 Papers based on the signalling game

Mu and Goodman [2021b], Ohmer et al. [2022], Yao et al. [2022], Rita et al. [2022a], Ohmer et al. [2021], Łukasz Kuciński et al. [2021], Portelance et al. [2021], Tucker et al. [2021], Dessì et al. [2021], Bullard et al. [2021], Perkins [2021a], Mihai and Hare [2021a], Denamganaï and Walker [2020], Guo et al. [2020], Li et al. [2020], Rita et al. [2020], Chowdhury et al. [2020a,b], Lan et al. [2020], Chaabouni et al. [2020], Luna et al. [2020], Kharitonov and Baroni [2020a], Ren et al. [2020], Słowik et al. [2020], Lowe et al. [2020], Keresztury and Bruni [2020], Dagan et al. [2020], Mihai and Hare [2019], Dessì et al. [2019], Guo et al. [2019], Steinert-Threlkeld [2019], Li and Bowling [2019], Kharitonov et al. [2020], Chaabouni et al. [2019a], Khomtchouk and Sudhakaran [2018], Bouchacourt and Baroni [2018], Lazaridou et al. [2018b], Havrylov

and Titov [2017], Lazaridou et al. [2016], Mahaut et al. [2023], Carmeli et al. [2022], Rita et al. [2022b], Downey et al. [2023]

A.4 Per system analysis

See Tables A.1 to A.4.

name	Token Count	Line Count	Tokens per Line	Tokens per Line SD
babyai-sr/GoToObj	130648	6116	21.361674	12.470737
babyai-sr/GoToObjLocked	272712	5629	48.447682	15.939260
babyai-sr/GoToObjLocked_ambiguous	229504	5507	41.674959	17.414703
babyai-sr/GoToObjLocked_ambiguous-freq_1	2605112	5179	503.014482	45.179870
babyai-sr/GoToObjLocked_ambiguous-freq_2	1061520	5396	196.723499	70.458489
babyai-sr/GoToObjLocked_ambiguous-freq_32	67248	5496	12.235808	3.993043
babyai-sr/GoToObjLocked_ambiguous-freq_4	402248	5728	70.224860	30.849604
babyai-sr/GoToObjLocked_ambiguous-msg_16	511840	5514	92.825535	33.765546
babyai-sr/GoToObjLocked_ambiguous-msg_32	855744	5508	155.363834	58.659355
babyai-sr/GoToObjLocked_ambiguous-msg_4	103228	5730	18.015358	6.603910
babyai-sr/GoToObjUnlocked	118752	6077	19.541221	7.060342
babyai-sr/GoToObjUnlocked-freq_1	1666456	6006	277.465201	205.399768
babyai-sr/GoToObjUnlocked-freq_2	333552	5777	57.737926	28.293252
babyai-sr/GoToObjUnlocked-freq_32	48616	6001	8.101316	0.894576
babyai-sr/GoToObjUnlocked-freq_4	193176	5762	33.525859	13.813609
babyai-sr/GoToObjUnlocked-msg_16	273008	6038	45.214972	18.173718
babyai-sr/GoToObjUnlocked-msg_32	469440	5765	81.429315	33.131090
babyai-sr/GoToObjUnlocked-msg_4	58588	5759	10.173294	4.351285
corpus-transfer-yao-et-al/cc	42977805	2865187	15.000000	0.000000
corpus-transfer-yao-et-al/coco_2014	1241745	82783	15.000000	0.000000
ec-at-scale/imagenet-10x1	2500000	250000	10.000000	0.000000
ec-at-scale/imagenet-10x10	2500000	250000	10.000000	0.000000
ec-at-scale/imagenet-1x1	2500000	250000	10.000000	0.000000
ec-at-scale/imagenet-1x10	2500000	250000	10.000000	0.000000
ec-at-scale/imagenet-5x5	2500000	250000	10.000000	0.000000
egg-discrimination/4-attr_4-val_3-dist_0-seed	110000	10000	11.000000	0.000000
egg-discrimination/4-attr_4-val_3-dist_1-seed	110000	10000	11.000000	0.000000
egg-discrimination/4-attr_4-val_3-dist_2-seed	110000	10000	11.000000	0.000000
egg-discrimination/6-attr_6-val_3-dist_0-seed	110000	10000	11.000000	0.000000
egg-discrimination/6-attr_6-val_3-dist_1-seed	110000	10000	11.000000	0.000000
egg-discrimination/6-attr_6-val_3-dist_2-seed	110000	10000	11.000000	0.000000
egg-discrimination/6-attr_6-val_9-dist_0-seed	110000	10000	11.000000	0.000000
egg-discrimination/6-attr_6-val_9-dist_1-seed	110000	10000	11.000000	0.000000
egg-discrimination/6-attr_6-val_9-dist_2-seed	110000	10000	11.000000	0.000000
egg-discrimination/8-attr_8-val_3-dist_0-seed	110000	10000	11.000000	0.000000
egg-discrimination/8-attr_8-val_3-dist_1-seed	110000	10000	11.000000	0.000000
egg-discrimination/8-attr_8-val_3-dist_2-seed	110000	10000	11.000000	0.000000
egg-reconstruction/4-attr_4-val_10-vocab_10-len	110000	10000	11.000000	0.000000
egg-reconstruction/6-attr_6-val_10-vocab_10-len	110000	10000	11.000000	0.000000
egg-reconstruction/8-attr_8-val_10-vocab_10-len	110000	10000	11.000000	0.000000
generalizations-mu-goodman/cub-concept	1333330	133333	10.000000	0.000000
generalizations-mu-goodman/cub-reference	1333330	133333	10.000000	0.000000
generalizations-mu-goodman/cub-set_reference	1333330	133333	10.000000	0.000000
generalizations-mu-goodman/shapeworld-concept	1164800	166400	7.000000	0.000000
generalizations-mu-goodman/shapeworld-reference	1164800	166400	7.000000	0.000000
generalizations-mu-goodman/shapeworld-set_reference	1164800	166400	7.000000	0.000000
nav-to-center/lexicon_size_11	65528	10000	6.552800	2.521193
nav-to-center/lexicon_size_118	58664	10000	5.866400	2.167199
nav-to-center/lexicon_size_17	59936	10000	5.993600	2.282533
nav-to-center/lexicon_size_174	63129	10000	6.312900	2.434747
nav-to-center/lexicon_size_25	61659	10000	6.165900	2.323010
nav-to-center/lexicon_size_255	60054	10000	6.005400	2.263000
nav-to-center/lexicon_size_37	62753	10000	6.275300	2.396604
nav-to-center/lexicon_size_54	58778	10000	5.877800	2.197195
nav-to-center/lexicon_size_7	61295	10000	6.129500	2.315368
nav-to-center/lexicon_size_80	60250	10000	6.025000	2.256097
nav-to-center/temperature_0.1	74939	10000	7.493900	3.180623
nav-to-center/temperature_0.167	72255	10000	7.225500	2.995171
nav-to-center/temperature_0.278	75732	10000	7.573200	3.106580
nav-to-center/temperature_0.464	79810	10000	7.981000	3.564778
nav-to-center/temperature_0.774	65665	10000	6.566500	2.527326
nav-to-center/temperature_1.29	62566	10000	6.256600	2.364647
nav-to-center/temperature_10	63105	10000	6.310500	2.397059
nav-to-center/temperature_2.15	62019	10000	6.201900	2.314160
nav-to-center/temperature_3.59	58786	10000	5.878600	2.187661
nav-to-center/temperature_5.99	61106	10000	6.110600	2.289491
rlupus/21-player.run-0	7131411	1001	7124.286713	445.837385
rlupus/21-player.run-1	7196469	999	7203.672673	396.806665
rlupus/21-player.run-2	7212723	1000	7212.723000	404.660045
rlupus/9-player.run-0	565164	1003	563.473579	14.708875
rlupus/9-player.run-1	417924	1010	413.786139	124.676603
rlupus/9-player.run-2	414612	1000	414.612000	124.794461
rlupus/9-player.run-3	416538	1003	415.292124	124.887337

Table A.1

name	Unique Tokens	Unique Lines	EoS Token Present	EoS Padding
babyai-sr/GoToObj	5	653	False	False
babyai-sr/GoToObjLocked	6	788	False	False
babyai-sr/GoToObjLocked_ambiguous	6	1253	False	False
babyai-sr/GoToObjLocked_ambiguous-freq_1	5	5171	False	False
babyai-sr/GoToObjLocked_ambiguous-freq_2	4	3078	False	False
babyai-sr/GoToObjLocked_ambiguous-freq_32	3	18	False	False
babyai-sr/GoToObjLocked_ambiguous-freq_4	6	3241	False	False
babyai-sr/GoToObjLocked_ambiguous-msg_16	9	4428	False	False
babyai-sr/GoToObjLocked_ambiguous-msg_32	3	1887	False	False
babyai-sr/GoToObjLocked_ambiguous-msg_4	2	1362	False	False
babyai-sr/GoToObjUnlocked	7	521	False	False
babyai-sr/GoToObjUnlocked-freq_1	7	4614	False	False
babyai-sr/GoToObjUnlocked-freq_2	8	3820	False	False
babyai-sr/GoToObjUnlocked-freq_32	4	41	False	False
babyai-sr/GoToObjUnlocked-freq_4	7	2766	False	False
babyai-sr/GoToObjUnlocked-msg_16	13	1740	False	False
babyai-sr/GoToObjUnlocked-msg_32	15	1430	False	False
babyai-sr/GoToObjUnlocked-msg_4	3	400	False	False
corpus-transfer-yao-et-al/cc	391	309405	True	True
corpus-transfer-yao-et-al/coco_2014	902	82783	True	False
ec-at-scale/imagenet-10x1	20	161235	False	False
ec-at-scale/imagenet-10x10	20	126775	False	False
ec-at-scale/imagenet-1x1	20	145834	False	False
ec-at-scale/imagenet-1x10	20	120182	False	False
ec-at-scale/imagenet-5x5	20	169505	False	False
egg-discrimination/4-attr_4-val_3-dist_0-seed	10	240	True	True
egg-discrimination/4-attr_4-val_3-dist_1-seed	10	220	True	True
egg-discrimination/4-attr_4-val_3-dist_2-seed	9	187	True	True
egg-discrimination/6-attr_6-val_3-dist_0-seed	8	2326	True	True
egg-discrimination/6-attr_6-val_3-dist_1-seed	10	3279	True	True
egg-discrimination/6-attr_6-val_3-dist_2-seed	9	1976	True	True
egg-discrimination/6-attr_6-val_9-dist_0-seed	9	2883	True	True
egg-discrimination/6-attr_6-val_9-dist_1-seed	9	1015	False	False
egg-discrimination/6-attr_6-val_9-dist_2-seed	10	2499	True	True
egg-discrimination/8-attr_8-val_3-dist_0-seed	10	2610	True	True
egg-discrimination/8-attr_8-val_3-dist_1-seed	10	2789	True	True
egg-discrimination/8-attr_8-val_3-dist_2-seed	9	2656	True	True
egg-reconstruction/4-attr_4-val_10-vocab_10-len	7	228	True	True
egg-reconstruction/6-attr_6-val_10-vocab_10-len	8	1373	True	True
egg-reconstruction/8-attr_8-val_10-vocab_10-len	8	1464	False	False
generalizations-mu-goodman/cub-concept	23	27163	False	False
generalizations-mu-goodman/cub-reference	23	39457	False	False
generalizations-mu-goodman/cub-set_reference	23	35042	False	False
generalizations-mu-goodman/shapeworld-concept	17	12481	False	False
generalizations-mu-goodman/shapeworld-reference	17	7683	False	False
generalizations-mu-goodman/shapeworld-set_reference	17	28061	True	False
nav-to-center/lexicon_size_11	8	2317	False	False
nav-to-center/lexicon_size_118	61	4392	False	False
nav-to-center/lexicon_size_17	15	3124	False	False
nav-to-center/lexicon_size_174	40	3226	False	False
nav-to-center/lexicon_size_25	12	1961	False	False
nav-to-center/lexicon_size_255	37	3706	False	False
nav-to-center/lexicon_size_37	22	2440	False	False
nav-to-center/lexicon_size_54	43	4911	False	False
nav-to-center/lexicon_size_7	7	1937	False	False
nav-to-center/lexicon_size_80	35	3486	False	False
nav-to-center/temperature_0.1	4	1437	False	False
nav-to-center/temperature_0.167	4	1313	False	False
nav-to-center/temperature_0.278	10	1308	False	False
nav-to-center/temperature_0.464	4	1498	False	False
nav-to-center/temperature_0.774	7	1639	False	False
nav-to-center/temperature_1.29	9	2100	False	False
nav-to-center/temperature_10	64	8793	False	False
nav-to-center/temperature_2.15	64	8643	False	False
nav-to-center/temperature_3.59	64	9044	False	False
nav-to-center/temperature_5.99	64	9263	False	False
rlupus/21-player.run-0	21	1001	False	False
rlupus/21-player.run-1	21	999	False	False
rlupus/21-player.run-2	21	1000	False	False
rlupus/9-player.run-0	9	1003	False	False
rlupus/9-player.run-1	9	1010	False	False
rlupus/9-player.run-2	9	1000	False	False
rlupus/9-player.run-3	9	1003	False	False

Table A.2

name	1-gram Entropy	1-gram Normalized Entropy	Entropy per Line
babyai-sr/GoToObj	1.237631	0.533019	26.437867
babyai-sr/GoToObjLocked	0.986990	0.381820	47.817369
babyai-sr/GoToObjLocked_ambiguous	1.724020	0.666942	71.848479
babyai-sr/GoToObjLocked_ambiguous-freq_1	1.463654	0.630362	736.239281
babyai-sr/GoToObjLocked_ambiguous-freq_2	1.385921	0.692961	272.643237
babyai-sr/GoToObjLocked_ambiguous-freq_32	0.358125	0.225952	4.381954
babyai-sr/GoToObjLocked_ambiguous-freq_4	1.996955	0.772528	140.235868
babyai-sr/GoToObjLocked_ambiguous-msg_16	2.555153	0.806061	237.183454
babyai-sr/GoToObjLocked_ambiguous-msg_32	1.350560	0.852108	209.828108
babyai-sr/GoToObjLocked_ambiguous-msg_4	0.922138	0.922138	16.612643
babyai-sr/GoToObjUnlocked	1.993155	0.709976	38.948688
babyai-sr/GoToObjUnlocked-freq_1	0.896426	0.319313	248.726924
babyai-sr/GoToObjUnlocked-freq_2	2.116083	0.705361	122.178237
babyai-sr/GoToObjUnlocked-freq_32	1.643569	0.821785	13.315074
babyai-sr/GoToObjUnlocked-freq_4	2.165184	0.771254	72.589650
babyai-sr/GoToObjUnlocked-msg_16	2.608207	0.704837	117.930014
babyai-sr/GoToObjUnlocked-msg_32	2.940985	0.752769	239.482412
babyai-sr/GoToObjUnlocked-msg_4	1.453225	0.916883	14.784087
corpus-transfer-yao-et-al/cc	1.398306	0.162386	20.974592
corpus-transfer-yao-et-al/coco_2014	6.599321	0.672235	98.989817
ec-at-scale/imagenet-10x1	3.980879	0.921089	39.808790
ec-at-scale/imagenet-10x10	3.908713	0.904391	39.087127
ec-at-scale/imagenet-1x1	4.121796	0.953694	41.217964
ec-at-scale/imagenet-1x10	3.975498	0.919844	39.754975
ec-at-scale/imagenet-5x5	4.213196	0.974842	42.131963
egg-discrimination/4-attr_4-val_3-dist_0-seed	2.996740	0.902109	32.964144
egg-discrimination/4-attr_4-val_3-dist_1-seed	2.494699	0.750979	27.441685
egg-discrimination/4-attr_4-val_3-dist_2-seed	2.564778	0.809097	28.212561
egg-discrimination/6-attr_6-val_3-dist_0-seed	2.581470	0.860490	28.396171
egg-discrimination/6-attr_6-val_3-dist_1-seed	2.887394	0.869192	31.761330
egg-discrimination/6-attr_6-val_3-dist_2-seed	2.573849	0.811959	28.312341
egg-discrimination/6-attr_6-val_9-dist_0-seed	2.861929	0.902838	31.481224
egg-discrimination/6-attr_6-val_9-dist_1-seed	2.462500	0.776832	27.087504
egg-discrimination/6-attr_6-val_9-dist_2-seed	2.750845	0.828087	30.259294
egg-discrimination/8-attr_8-val_3-dist_0-seed	2.426752	0.730525	26.694277
egg-discrimination/8-attr_8-val_3-dist_1-seed	2.556315	0.769528	28.119469
egg-discrimination/8-attr_8-val_3-dist_2-seed	2.802140	0.883977	30.823535
egg-reconstruction/4-attr_4-val_10-vocab_10-len	2.296329	0.817969	25.259614
egg-reconstruction/6-attr_6-val_10-vocab_10-len	2.573243	0.857748	28.305674
egg-reconstruction/8-attr_8-val_10-vocab_10-len	2.295767	0.765256	25.253441
generalizations-mu-goodman/cub-concept	3.752944	0.829644	37.529443
generalizations-mu-goodman/cub-reference	3.103881	0.686159	31.038812
generalizations-mu-goodman/cub-set_reference	3.213538	0.710400	32.135376
generalizations-mu-goodman/shapeworld-concept	3.226724	0.789420	22.587066
generalizations-mu-goodman/shapeworld-reference	3.224439	0.788861	22.571074
generalizations-mu-goodman/shapeworld-set_reference	3.365556	0.823385	23.558893
nav-to-center/lexicon_size_11	2.805418	0.935139	18.383341
nav-to-center/lexicon_size_118	3.767532	0.635255	22.101847
nav-to-center/lexicon_size_17	3.186153	0.815521	19.096524
nav-to-center/lexicon_size_174	3.245330	0.609803	20.487443
nav-to-center/lexicon_size_25	2.804201	0.782212	17.290421
nav-to-center/lexicon_size_255	3.534679	0.678513	21.227163
nav-to-center/lexicon_size_37	3.028477	0.679117	19.004602
nav-to-center/lexicon_size_54	3.754792	0.691966	22.069917
nav-to-center/lexicon_size_7	2.758577	0.982625	16.908697
nav-to-center/lexicon_size_80	3.457586	0.674088	20.831957
nav-to-center/temperature_0.1	1.994309	0.997155	14.945156
nav-to-center/temperature_0.167	1.981753	0.990876	14.319155
nav-to-center/temperature_0.278	1.986637	0.598037	15.045198
nav-to-center/temperature_0.464	1.982692	0.991346	15.823868
nav-to-center/temperature_0.774	2.311150	0.823248	15.176164
nav-to-center/temperature_1.29	2.754878	0.869067	17.236170
nav-to-center/temperature_10	4.905167	0.817528	30.954056
nav-to-center/temperature_2.15	4.966695	0.827782	30.802945
nav-to-center/temperature_3.59	5.340638	0.890106	31.395475
nav-to-center/temperature_5.99	5.266347	0.877724	32.180539
rlupus/21-player.run-0	4.062520	0.924915	28942.558214
rlupus/21-player.run-1	4.196960	0.955523	30233.522552
rlupus/21-player.run-2	3.997152	0.910033	28830.352466
rlupus/9-player.run-0	3.079577	0.971498	1735.260160
rlupus/9-player.run-1	3.119583	0.984119	1290.840311
rlupus/9-player.run-2	3.090164	0.974838	1281.218984
rlupus/9-player.run-3	3.111235	0.981485	1292.071343

Table A.3

name	2-gram Entropy	2-gram Conditional Entropy
babyai-sr/GoToObj	1.544519	0.306888
babyai-sr/GoToObjLocked	1.147285	0.160295
babyai-sr/GoToObjLocked_ambiguous	1.978413	0.254393
babyai-sr/GoToObjLocked_ambiguous-freq_1	2.071162	0.607508
babyai-sr/GoToObjLocked_ambiguous-freq_2	1.538991	0.153070
babyai-sr/GoToObjLocked_ambiguous-freq_32	0.420175	0.062050
babyai-sr/GoToObjLocked_ambiguous-freq_4	2.406197	0.409242
babyai-sr/GoToObjLocked_ambiguous-msg_16	3.097571	0.542418
babyai-sr/GoToObjLocked_ambiguous-msg_32	1.463220	0.112660
babyai-sr/GoToObjLocked_ambiguous-msg_4	1.717505	0.795367
babyai-sr/GoToObjUnlocked	2.497606	0.504450
babyai-sr/GoToObjUnlocked-freq_1	1.092966	0.196541
babyai-sr/GoToObjUnlocked-freq_2	2.877898	0.761815
babyai-sr/GoToObjUnlocked-freq_32	1.731359	0.087790
babyai-sr/GoToObjUnlocked-freq_4	2.979210	0.814026
babyai-sr/GoToObjUnlocked-msg_16	3.043978	0.435771
babyai-sr/GoToObjUnlocked-msg_32	3.157215	0.216230
babyai-sr/GoToObjUnlocked-msg_4	2.307255	0.854029
corpus-transfer-yao-et-al/cc	2.059689	0.661383
corpus-transfer-yao-et-al/coco_2014	12.884451	6.285130
ec-at-scale/imagenet-10x1	6.811992	2.831113
ec-at-scale/imagenet-10x10	6.328754	2.420041
ec-at-scale/imagenet-1x1	6.882813	2.761016
ec-at-scale/imagenet-1x10	6.375876	2.400379
ec-at-scale/imagenet-5x5	7.137788	2.924591
egg-discrimination/4-attr_4-val_3-dist_0-seed	4.434835	1.438094
egg-discrimination/4-attr_4-val_3-dist_1-seed	3.550278	1.055580
egg-discrimination/4-attr_4-val_3-dist_2-seed	3.544613	0.979835
egg-discrimination/6-attr_6-val_3-dist_0-seed	3.917021	1.335551
egg-discrimination/6-attr_6-val_3-dist_1-seed	4.308021	1.420628
egg-discrimination/6-attr_6-val_3-dist_2-seed	3.738390	1.164541
egg-discrimination/6-attr_6-val_9-dist_0-seed	4.371053	1.509123
egg-discrimination/6-attr_6-val_9-dist_1-seed	3.578326	1.115826
egg-discrimination/6-attr_6-val_9-dist_2-seed	4.070906	1.320061
egg-discrimination/8-attr_8-val_3-dist_0-seed	3.504384	1.077631
egg-discrimination/8-attr_8-val_3-dist_1-seed	3.712531	1.156216
egg-discrimination/8-attr_8-val_3-dist_2-seed	4.006086	1.203946
egg-reconstruction/4-attr_4-val_10-vocab_10-len	3.212115	0.915787
egg-reconstruction/6-attr_6-val_10-vocab_10-len	3.750294	1.177051
egg-reconstruction/8-attr_8-val_10-vocab_10-len	3.515011	1.219244
generalizations-mu-goodman/cub-concept	5.686797	1.933852
generalizations-mu-goodman/cub-reference	5.641346	2.537465
generalizations-mu-goodman/cub-set_reference	5.509904	2.296366
generalizations-mu-goodman/shapeworld-concept	6.040857	2.814134
generalizations-mu-goodman/shapeworld-reference	5.908455	2.684016
generalizations-mu-goodman/shapeworld-set_reference	6.409305	3.043749
nav-to-center/lexicon_size_11	4.240224	1.434806
nav-to-center/lexicon_size_118	5.389004	1.621472
nav-to-center/lexicon_size_17	4.655472	1.469319
nav-to-center/lexicon_size_174	4.717891	1.472561
nav-to-center/lexicon_size_25	4.106729	1.302528
nav-to-center/lexicon_size_255	5.098629	1.563950
nav-to-center/lexicon_size_37	4.335838	1.307361
nav-to-center/lexicon_size_54	5.463441	1.708649
nav-to-center/lexicon_size_7	4.123176	1.364599
nav-to-center/lexicon_size_80	5.001934	1.544348
nav-to-center/temperature_0.1	3.405157	1.410848
nav-to-center/temperature_0.167	3.469046	1.487293
nav-to-center/temperature_0.278	3.396763	1.410126
nav-to-center/temperature_0.464	3.377160	1.394467
nav-to-center/temperature_0.774	3.777791	1.466642
nav-to-center/temperature_1.29	4.202502	1.447624
nav-to-center/temperature_10	8.121348	3.216181
nav-to-center/temperature_2.15	7.739814	2.773120
nav-to-center/temperature_3.59	8.433494	3.092856
nav-to-center/temperature_5.99	8.660965	3.394618
rlupus/21-player.run-0	6.956412	2.893892
rlupus/21-player.run-1	7.403071	3.206111
rlupus/21-player.run-2	7.039882	3.042730
rlupus/9-player.run-0	5.883233	2.803656
rlupus/9-player.run-1	5.925070	2.805487
rlupus/9-player.run-2	5.979073	2.888910
rlupus/9-player.run-3	5.865222	2.753987

Table A.4

Appendix B

XferBench

B.1 Hyperparameters

Causal language modeling

For values not listed, see Hugging Face Transformers' defaults at `https://huggingface.co/docs/transformers/v4.36.1/en//model_doc/gpt2#transformers.GPT2Config`.

- Model: GPT-2
- Tokenizer: Byte pair encoding
- Hidden size: 768 (default)
- Vocabulary size: 30 000
- Context length: 256
- Number of layers: 6
- Number of attention heads: 6
- Learning rate: $1 \cdot 10^{-4}$
- Optimizer: AdamW
- Weight decay: 0.01
- Learning rate schedule: linear (to 0)
- Batch size: 32
- Train dataset size: $15 \cdot 10^6$ tokens
- Train epochs: 5
- Tune dataset size: $2 \cdot 10^6$ tokens
- Train epochs: 10

Machine translation

For values not listed, see Hugging Face Transformers' defaults at `https://huggingface.co/docs/transformers/v4.36.1/en/model_doc/bart#transformers.BartConfig`. The following is for the *Full* setting.

- Model: BART
- Training objective: text infilling only (see note below)
- Tokenizer: Byte pair encoding
- Hidden size: 512

- Vocabulary size: 30 000
- Context length: 512
- Number of encoder layers: 6
- Number of decoder layers: 6
- Number of encoder attention heads: 8
- Number of decoder attention heads: 8
- Encoder feedforward dimension: 2048
- Decoder feedforward dimension: 2048
- Train learning rate: $1 \cdot 10^{-4}$
- Tune learning rate: $2 \cdot 10^{-4}$
- Optimizer: AdamW
- Weight decay: 0.01
- Learning rate schedule: linear (to 0)
- Batch size: 32
- Train dataset size: $100 \cdot 10^6$ tokens
- Train epochs: 5
- Tune dataset size: $50 \cdot 10^6$ tokens
- Train epochs: 3
- Test beam size: $1, 3, 5$ (final metric averaged across each size)
- Test context size: 128

The objective used to pretrain BART was text infilling *only*; we cannot use the sentence permutation objective because we do not know *a priori* what constitutes a sentence in an emergent language, hence we do not use it for any settings. For the *Frozen* setting, all is as above, but all non-embedding layers are frozen for the duration of tuning. For the *Reduced* setting, all is as above except for the following:

- Tune learning rate: $1 \cdot 10^{-5}$
- Tune dataset size: $10 \cdot 10^6$

Generic signalling game

We use the following hyperparameters for the *Disc, small* emergent language.

- Game (from EGG): `egg.zoo.basic_games.play`
- Message optimization: Gumbel-softmax (as opposed to REINFORCE)
- Game type: discrimination
- Number of attributes: 4
- Number of values: 4
- Number of distractors: 5
- Vocabulary size: 6
- Max message length: 10
- Number of examples: 32 768
- Batch size; 1024
- Number of epochs: 10
- Sender hidden size: 256
- Receiver hidden size: 512
- Sender embedding size: 32

- Receiver embedding size: 32
- Sender network type: GRU
- Receiver network type: GRU
- Learning rate: 0.001

The *Disc, large* setting uses the same hyperparameters as above with the exception of the following.

- Number of attributes: 12
- Number of values: 8
- Number of distractors: 5
- Number of examples: $3.5 \cdot 10^6$
- Max message length: 30
- Vocabulary size: 100
- Number of epochs: 100

The *Recon, large* setting is as in *Disc, large* with the following changes.

- Game type: reconstruction
- Number of attributes: 8
- Number of distractors: N/A
- Number of examples: $1 \cdot 10^6$
- Number of epochs: 10

B.2 Example of benchmark input format

The input format for the benchmark is simple: integer arrays in a JSON format separated by newlines (i.e., JSON Lines, JSONL, `*.jsonl`). The following is an example of file contents in this format:

```
[3, 1, 4, 1, 5, 9, 2]
[6, 5, 3, 5, 8, 9, 7, 9, 3]
[2, 3, 8, 4]
[6, 2, 6, 4, 3, 3]
[8, 3, 2, 7, 9, 5, 0, 2, 8, 8, 4]
```

B.3 Computing resources used

See Table B.1 for rough estimates of the compute used in writing this paper. Most experiments were run on a shared cluster comprising approximately 150 NVIDIA A6000 (or comparable) GPUs.

B.4 Additional results

BLEU scores for machine translation

See Table B.2.

Item	Base GH	n items	Total
XferBench	6	45	270
MT	8	50	400
Other experiments	2	50	100
Total			770

Table B.1: Estimate of compute used for this paper in GPU-hours (specifically NVIDIA RTX 2080 Ti–hours).

Source	Full	Frozen	Reduced
French	12.93	5.33	6.61
Spanish	13.32	4.52	6.35
Russian	12.93	4.37	7.02
Chinese	12.71	3.04	6.03
Korean	12.83	2.95	6.36
Arabic	13.12	4.16	6.74
Hindi	12.72	3.20	5.24
Paren, real	12.60	0.65	6.26
Paren, synth	13.19	0.82	6.15
Disc, large	12.93	2.08	4.44
Disc, small	0.17	0.19	0.38
Rec, large	1.92	0.86	2.50
Yao+	0.01	1.04	2.57
Mu+, SW	0.00	1.05	1.86
Mu+, CUB	12.71	1.45	2.35
Random	0.00	0.00	1.02
No pretrain	0.10	0.06	3.43

Table B.2: BLEU scores for machine translation experiment. Colors normalized by column.

Raw cross-entropies on XferBench

See Table B.3.

Writing system matrix for normalized XferBench scores

See Tables B.5 and B.6. Scores for reach writing system are aggregated by taking the mean. Table B.4 gives the writing system classification for the languages used in the experiments. Although the class imbalance makes it impossible to draw any definitive claims, the preliminary results do not suggest any correlation in XferBench between the writing systems of the source and target languages.

Source	Danish	Basque	Persian	Finnish	Hebrew	Indonesian	Japanese	Kazakh	Romanian	Urdu	*Mean*
French	4.93	6.03	5.04	5.62	5.48	4.87	5.23	5.46	5.15	4.43	5.22
Spanish	4.92	6.06	5.03	5.61	5.47	4.82	5.25	5.46	5.12	4.42	5.22
Russian	4.94	6.04	5.04	5.65	5.48	4.88	5.27	5.48	5.14	4.45	5.24
Chinese	4.89	6.02	5.01	5.58	5.43	4.76	5.18	5.44	5.12	4.39	5.18
Korean	4.89	6.01	5.02	5.57	5.44	4.78	5.20	5.45	5.12	4.38	5.19
Arabic	4.90	6.02	5.02	5.59	5.45	4.81	5.22	5.44	5.13	4.40	5.20
Hindi	4.94	6.06	5.08	5.65	5.47	4.83	5.29	5.52	5.20	4.46	5.25
Paren, real	5.07	6.11	5.11	5.75	5.59	5.06	5.38	5.57	5.22	4.56	5.34
Paren, synth	5.08	6.13	5.14	5.74	5.58	5.09	5.43	5.58	5.26	4.57	5.36
Disc, large	5.00	6.06	5.11	5.71	5.52	4.92	5.34	5.56	5.25	4.49	5.30
Disc, small	5.09	6.06	5.17	5.80	5.59	5.05	5.41	5.65	5.31	4.56	5.37
Rec, large	5.09	6.06	5.16	5.79	5.57	5.04	5.41	5.64	5.30	4.55	5.36
Yao+	5.07	6.03	5.17	5.79	5.56	5.03	5.41	5.65	5.31	4.56	5.36
Mu+, SW	5.09	6.10	5.18	5.80	5.58	5.05	5.42	5.65	5.33	4.58	5.38
Mu+, CUB	5.08	6.06	5.18	5.79	5.58	5.05	5.42	5.65	5.32	4.56	5.37
Random	5.23	6.17	5.31	5.92	5.71	5.22	5.55	5.76	5.45	4.72	5.50
No pretrain	5.17	6.10	5.23	5.85	5.66	5.14	5.47	5.68	5.38	4.65	5.43
Mean	5.02	6.07	5.12	5.72	5.54	4.96	5.35	5.57	5.24	4.51	5.31

Table B.3: Cross-entropies across all source and target languages. Colors normalized by column.

Type	Writing System	Language
Abjad	Arabic	ar
		fa
		ur
	Hebrew	he
Abugida	Devanagari	hi
Alphabet	Cyrillic	kk
		ru
	Hangul	ko
	Latin	da
		es
		eu
		fi
		fr
		id
		ro
Logographic	Chinese	zh
Mixed	Japanese	ja

Table B.4: Coarse and fine classifications of writing systems of human languages (source and target) used in the experiments.

Source	Arabic	Cyrillic	Hebrew	Japanese	Latin
Arabic	−0.65	−0.81	−0.42	−0.35	−0.56
Chinese	−1.05	−0.93	−1.60	−1.46	−1.00
Cyrillic	0.63	0.68	1.07	0.90	0.78
Devanagari	1.62	1.93	0.39	1.47	1.23
Hangul	−0.98	−0.41	−0.89	−0.80	−1.04
Latin	0.22	−0.22	0.72	0.12	0.29

Table B.5: Normalized XferBench scores by writing system (lower is better). Color is normalized across all values.

Source	Abjad	Alphabet	Mixed
Abjad	−0.57	−0.60	−0.35
Abugida	1.21	1.35	1.47
Alphabet	0.15	0.06	0.09
Logographic	−1.23	−0.99	−1.46

Table B.6: Normalized XferBench scores by writing system type (lower is better). Color is normalized across all values.

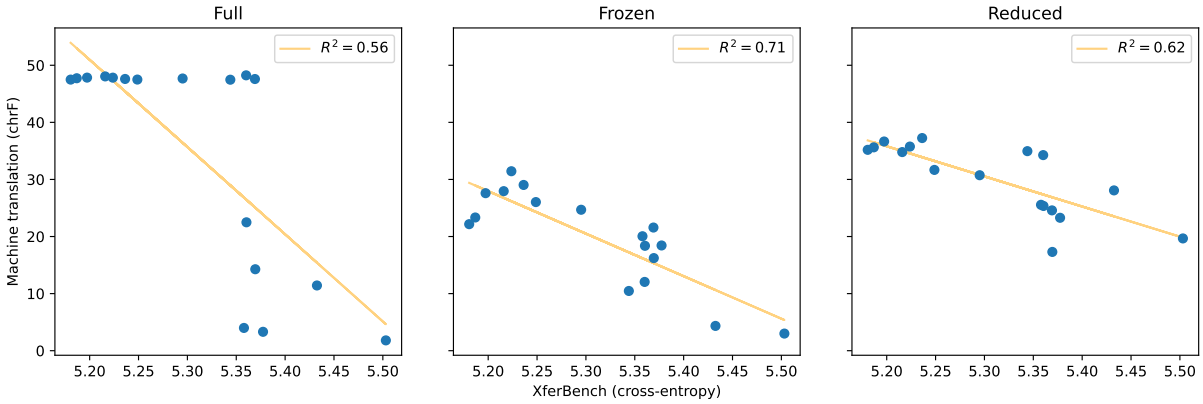


Figure B.1: Scatter plots showing XferBench score versus machine translation score.

Scatter plots for XferBench and MT

See Figure B.1.

B.5 Cross-entropy confidence interval computation

Let $s \in S$ and $t \in T$ represent source and target languages, respectively. $h_{s,t}$ represents the test cross-entropy of a model pretrained on s and evaluated on t. As sated in Equation (4.1),

the score on XferBench is the mean cross-entropy across all target languages:

$$h_s = \underset{t' \in T}{\text{mean}} \left(h_{s,t'} \right). \tag{B.1}$$

We would like to calculate a confidence interval (i.e., h_s^- and h_s^+) for a source language's mean cross-entropy using the different cross-entropies on the target languages (i.e., $h_{s,t}$ for $t \in T$), yet these samples are not i.i.d., since the mean of cross-entropy each *target* language can vary. Thus, if we would like to use bootstrapping to calculate confidence intervals, we must first normalize the cross-entropies. Let $\hat{h}_{s,t}$ be the normalized score:

$$\hat{h}_{s,t} = \frac{h_{s,t} - \text{mean}_{s' \in S} \left(h_{s',t} \right)}{\text{stdev}_{s' \in S} \left(h_{s',t} \right)}. \tag{B.2}$$

Given the normalized scores, we can now bootstrap in order to compute confidence intervals for $\hat{h}_s$ (i.e., in the normalized space).[1] Let $\hat{h}_s^+$ and $\hat{h}_s^-$ be the upper and lower bounds of the confidence interval computed using bootstrapping in the normalized space. We can now translate these back into the raw cross-entropy space using the means and standard deviations from before:

$$h_s^+ = \hat{h}_s^+ \cdot \underset{s' \in S}{\text{stdev}} \left(h_{s',t} \right) + \underset{s' \in S}{\text{mean}} \left(h_{s',t} \right) \tag{B.3}$$

$$h_s^- = \hat{h}_s^- \cdot \underset{s' \in S}{\text{stdev}} \left(h_{s',t} \right) + \underset{s' \in S}{\text{mean}} \left(h_{s',t} \right). \tag{B.4}$$

B.6 Error analysis

In the *Full* setting of the machine translation task, the *Yao+* and *Mu+, SW* settings perform worse than expected (*a priori* and compared to the other results in the setting). Validation loss converged while chrF and BLEU scores remained near zero. We provide a couple examples (taken from the predefined test set of WMT 2014) of model output to provide some insight into the reason for this. No post processing used, generation is capped at 50 tokens, and "\u0000" represent single non-printable characters.

Example 1 *Input*: "And while Congress can't agree on whether to proceed, several states are not waiting."

Reference: "Et tandis que les membres du Congrès n'arrivent pas à se mettre d'accord pour savoir s'il faut continuer, plusieurs États n'ont pas attendu."

[Model pretrained on] *French*: "#Et alors que le Congrès ne peut pas convenir de poursuivre, plusieurs États ne sont pas en attente. » (traduction libre) Le Parlement européen. Le Parlement européen est d'avis que le Parlement européen doit être en mesure de faire preuve#"

Disc, large: "#Et bien que le Congrès ne puisse pas convenir de la marche à suivre, plusieurs États ne sont pas en attente.\u2028\u2028[Traduit par la Rédaction]\u2028(Traduit par la Rédaction)\u2028(Tra#"

[1] This is not intended to be statistically rigorous. Our cross-entropies are unlikely to be normally distributed, but this still be helpful for generally gauging uncertainty.

Yao+: "#Annexe II, par.\xa02.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.#"

Mu+, SW: "#Annexe II.\xa0\xa0: Appendice I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I#"

Example 2 *Input*: "This really is a must for our nation."

Reference: "Cela est vraiment indispensable pour notre nation."

French: "#C'est vraiment une nécessité pour notre nation. Nous devons y parvenir. Nous devons y parvenir. Nous devons y parvenir. Nous devons y parvenir. Nous devons y parvenir. Nous devons y parvenir. Nous devons y#"

Disc, large: "#C'est vraiment un devoir pour notre nation. C'est un devoir.\u2028\u2028(...)\u2028\u2028(...)\u2028(...)\u2028(...)\u2028(...)\u2028(...)\u2028(#"

Yao+: "#Annexe II, par.\xa02.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.#"

Mu+, SW: "#Annexe II.\xa0\xa0: Appendice I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I.\xa0I#"

Discussion Although all of the models have trouble terminating properly, the *French* and *Disc, large* models (which have high chrF scores) clearly condition their generation on the text, whereas *Yao+* and *Mu+, SW* give the same output regardless of the input. Although this is unexpected, we can see in the *Full* setting in Figure B.1 that there is sharp drop off between high-performing and low-performing languages. We suspect that the higher learning rate during tuning caused this bimodal distribution of results and is at least in part responsible for the poor performance *Yao+* and *Mu+, SW* models on the MT experiment's *Full* setting.

Appendix C

Optimizing with Transfer Learning

C.1 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 C.1, we show the R^2 values derived from training a linear model on just one of

	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 C.1: 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.

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).

C.2 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 learning, 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. We did not vary the neural architecture beyond altering the number of units in the hidden and embedding layers; for example, we did not add additional layers, try different RNN cells (e.g., LSTM), or use transformers.

C.3 Full Table of Hyperparameters

In Table C.2, we show all of the hyperparameters selected for the searches and trials referenced in the paper.

C.4 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.

#	|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 C.2: 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.).

C.5 Synthetic Languages

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

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

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 sampled

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

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

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 "{}", "(())", and "[][]", which could then be interleaved into "{[}(()])".

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: 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.

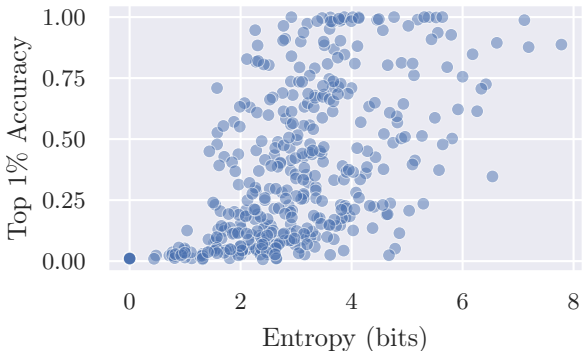


Figure C.1: Entropy versus accuracy for Search 5r.

C.6 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 Figure C.1. 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.

C.7 Hyperparameter Scatter Plots

Figures C.2 to C.5 show the univariate scatter plots for hyperparameter Searches 1–4. The y-axis is XferBench-da score (or some smaller variation 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.

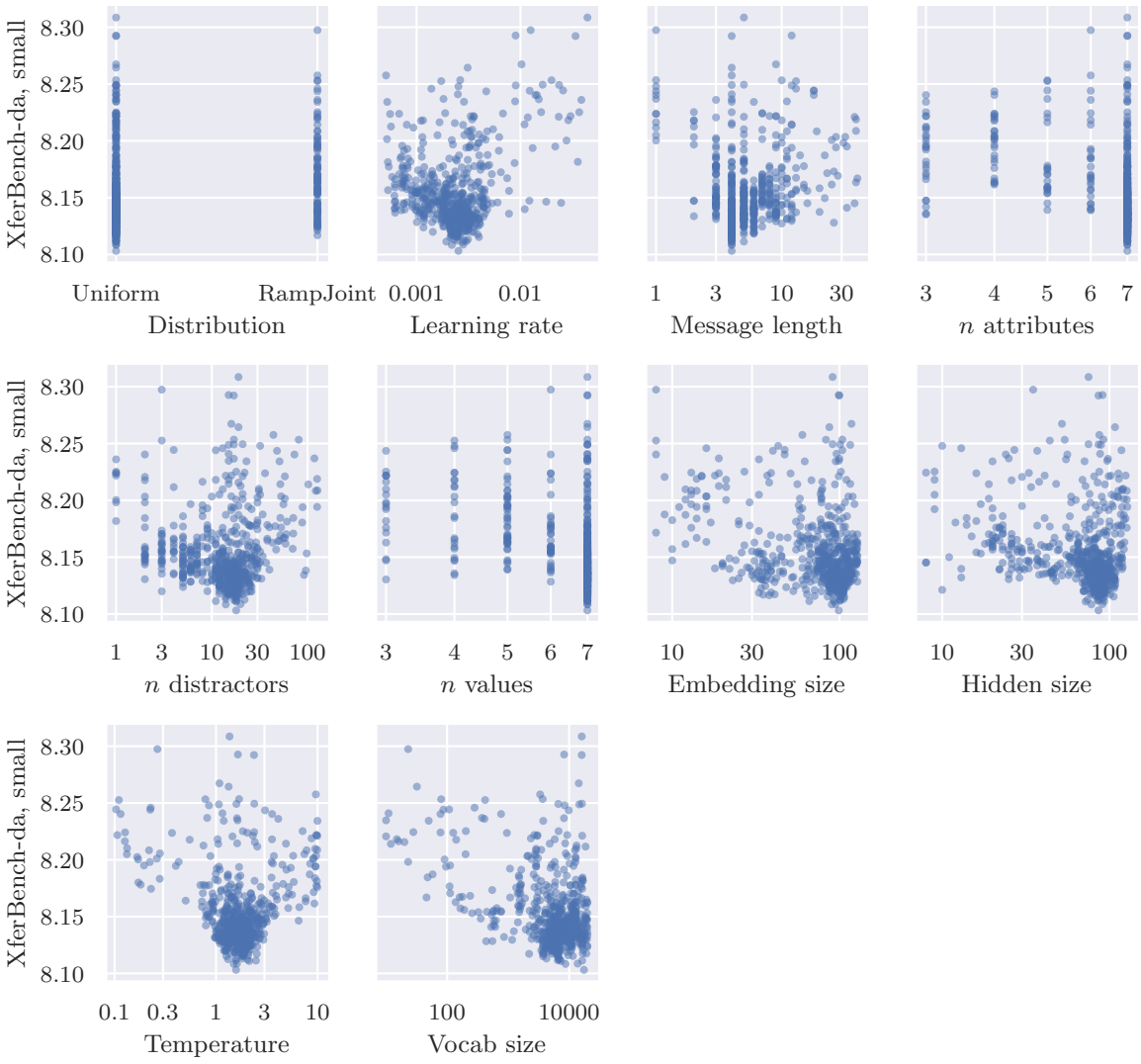


Figure C.2: Objective values for Search 1 by individual hyperparameter.

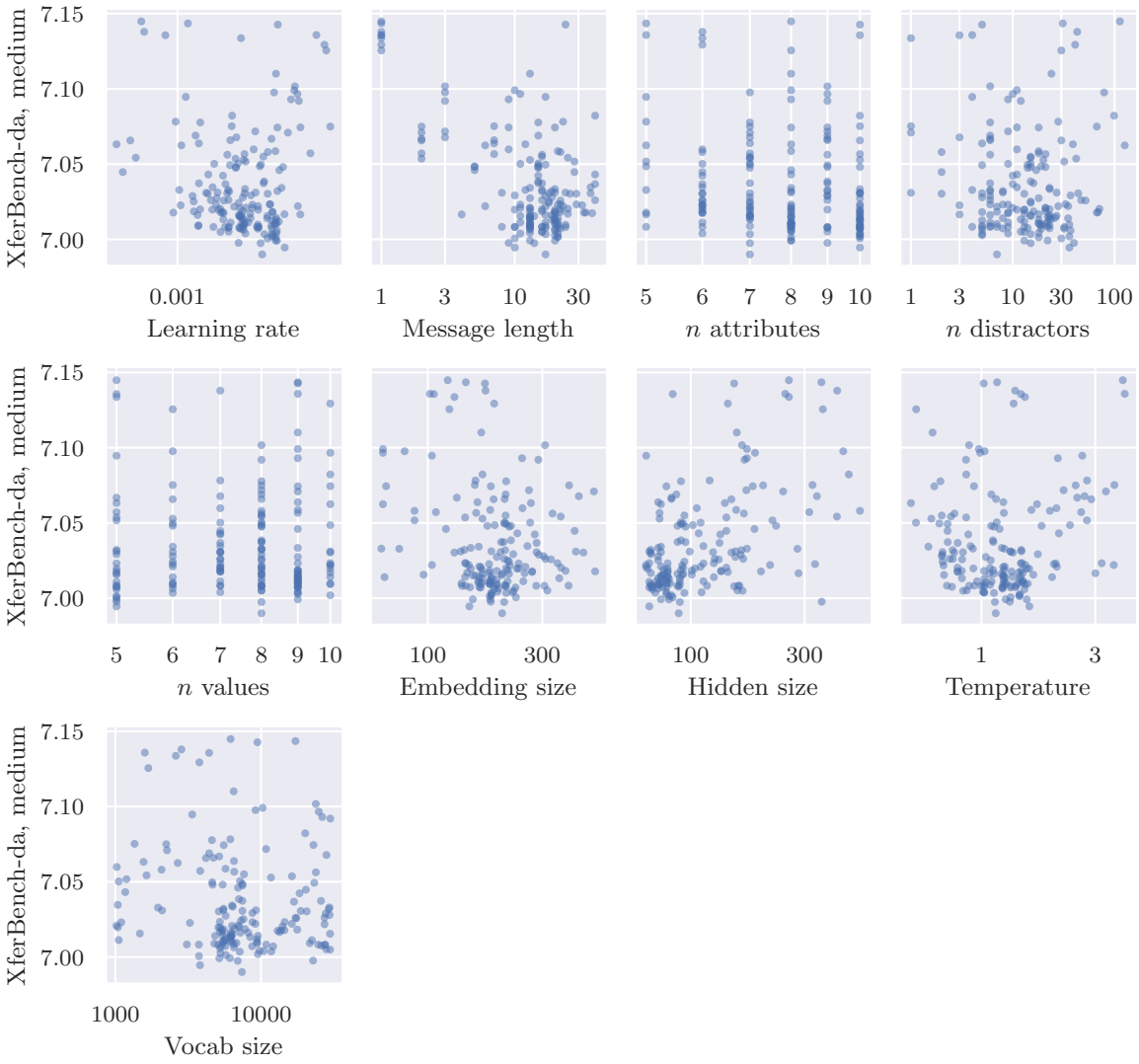


Figure C.3: Objective values for Search 2 by individual hyperparameter.

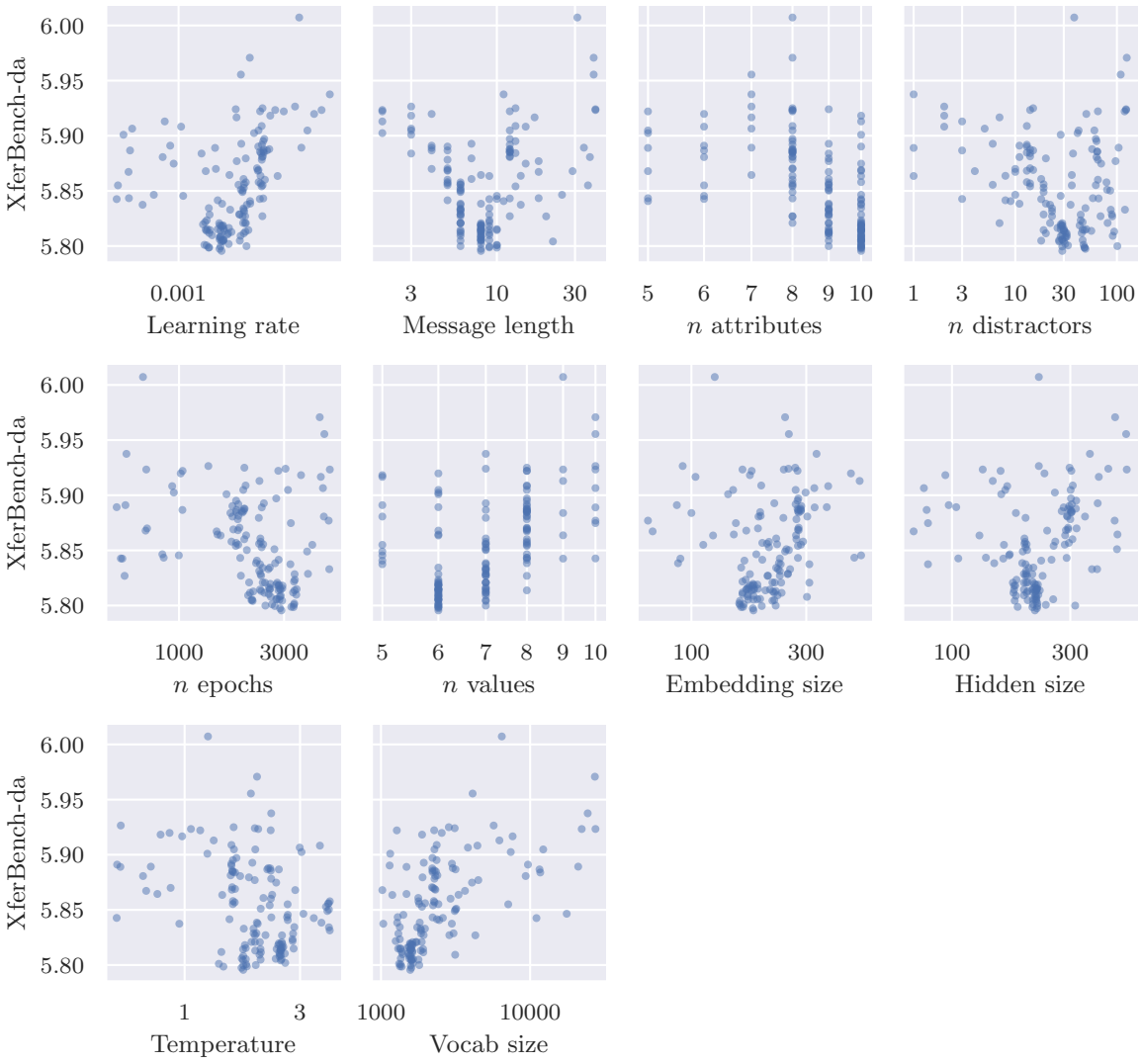


Figure C.4: Objective values for Search 3 by individual hyperparameter.

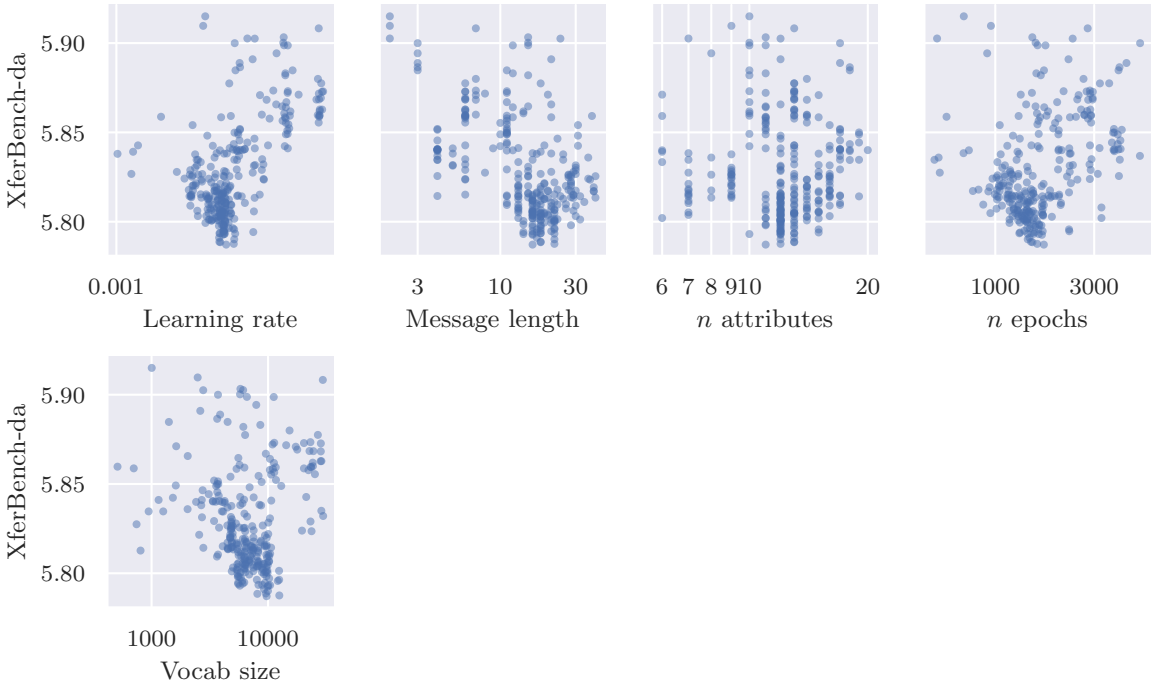


Figure C.5: Objective values for Search 4 by individual hyperparameter.

Appendix D

Morpheme Induction

D.1 Algorithm

Candidate generation

For simplicity's sake (and inductive bias), we limit the candidate generation functions to all non-empty substrings for forms and all non-empty subsets for meanings. Nevertheless, we could extend form candidate generation to non-contiguous forms to detect non-concatenative morphology (e.g., the form "x.z" matching "xyz" and "xwz"). In fact, we could could use arbitrary regular expressions to represent forms (or meanings) such as "^..x" or "x+" to represent absolute position and optional repetitions, respectively. We could consider empty forms and empty meanings to explicitly identify forms and meanings which do not have mappings (as opposed to implicitly not including them in the morphology).

Of course, part of the difficulty of extending the complexity of the candidate generation is that it expands the already (sometimes intractably) large search space. One method of making this tractable, though, is adding heuristics that determine which form candidates should be considered rather than considering every possible candidate.

Ambiguous pair application

In some cases of applying a morpheme to record in the dataset, there are multiple applications possible. Say we have the utterance "x y z x y" meaning $\{A, B\}$ and we want to apply the morpheme ("x y", $\{A\}$). The form matches two substrings in the utterance, so there are two possible ways to apply the morpheme. As a heuristic for selecting the best application, CSAR break ties by selecting the substring least likely to be a morpheme (as determined by the morpheme weights). Going back to the above example, if it is the case the morpheme ("z x y", $\{B\}$) has a higher weight than ("x y z", $\{B\}$), then CSAR will apply ("x y", $\{A\}$) to the first instance of "x y" instead of the second.

This search can be very computationally expensive since it can entail going through a large number of morpheme candidates. Thus for the experiments with human language data, we do not perform this search and select the best form quasirandomly.

Heuristic optimizations

Below we include a summary of heuristic optimizations available in CSAR:

max input records Only consider a certain number of records from the input data; 20 000 for machine translation, image captions, and ShapeWorld.

max inventory size Stop after inducing a certain number of morphemes; 300 for image captions and machine translation settings.

***n*-gram semantics** Treat complete meanings as ordered and generate meaning candidates identically to forms (i.e., as n-grams); used for machine translation data where the "meanings" are sentences.

max form/meaning size Only consider form/meaning candidates up to a certain size; 3 for machine translation (form and meaning) and image captions (form only), 2 for image captions meaning.

no search best form When ablating a form with multiple matches in an utterance, do not search for best form, simply choose it randomly; no search for image captions and machine translation.

form/meaning vocabulary size Only consider the most common form/meaning candidates; 100 000 for image captions and machine translation.

token vocabulary size Only consider the most common form/meaning tokens and ignore an form meaning candidates which contain an unknown token; 1000 for image captions and 500 for machine translation.

co-occurrence threshold Zero out any co-occurrences which fall below a certain threshold (e.g., if a form and meaning candidate only occur once, treat it as never co-occurring); 1 for ShapeWorld, 10 for image captions, and 100 for machine translation.

D.2 Empirical Validation

Procedural dataset hyperparameters

The following hyperparameters were used for generating the procedural datasets. Each dataset uses 4 attributes and 4 values except for the sparse setting which uses 8 independent values.

Synonymy $\{1, 3\}$; forms per meaning

Polysemy $\{0, 0.15\}$; proportion of meanings mapped to an already-used form

Multi-token forms $\{\{1\}, \{1, 2, 3, 4\}\}$; possible tokens per form

Vocab size $\{10, 50\}$; only applies to non-unity multi-token forms

Sparse meanings $\{\text{true}, \text{false}\}$

Distribution imbalance $\{\text{true}, \text{false}\}$; non-uniform distribution is based on the ramp function, i.e., probability of given value for an attribute is proportional to its index $+ 1$.

Dataset size $\{50, 500\}$

Noise forms $\{0, 0.5\}$; $1 - p$ of parameter of geometric distribution

Shuffle form $\{\text{true}, \text{false}\}$

Non-compositionality $\{\text{true}, \text{false}\}$

Random seeds 3 per hyperparameter setting

	CSAR	IBM Model 1	IBM Model 3	Morfessor	BPE	ULM	Records
Exact F_1, form	0.868	0.616	0.595	0.827	0.624	0.670	0.133
Fuzzy F_1, form	0.960	0.899	0.893	0.949	0.890	0.891	0.637
Fuzzy prec., form	0.954	0.855	0.850	0.933	0.852	0.853	0.597
Fuzzy recall, form	0.967	0.952	0.946	0.967	0.934	0.938	0.701
Exact F_1	0.788	0.375	0.379	0.000	0.000	0.000	0.101
Fuzzy F_1	0.899	0.721	0.726	0.000	0.000	0.000	0.441
Fuzzy prec.	0.881	0.641	0.640	0.000	0.000	0.000	0.390
Fuzzy recall	0.921	0.855	0.866	0.000	0.000	0.000	0.543

Table D.1: Results of baseline methods on the procedural datasets.

Non-unity polysemy and synonymy rates for the non-compositional dataset implementation were not implemented and are excluded from the above grid.

Tokenizer vocabulary size

The heuristic for the tokenizer vocabulary size is as follows:

$$|V| = \left\lfloor \frac{|\mathcal{T}_{\text{meaning}}|}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{|r_{\text{form}}|}{|r_{\text{meaning}}|} \right\rfloor + |\mathcal{T}_{\text{form}}|, \tag{D.1}$$

where $\mathcal{T}_{\text{meaning}}$ is the set of all meaning tokens in the dataset (likewise for $\mathcal{T}_{\text{form}}$), $\mathcal{R}$ is the multiset of records in dataset, r_{form} is the particular form (utterance) for an individual record (likewise for r_{meaning}. This heuristic can be interpreted as the mean form tokens per meaning tokens times the number unique meaning tokens added to the number of unique form tokens (since each of them will automatically be included in the vocabulary).

Additional procedural dataset results

Table D.1 shows all results of baseline methods on the procedural datasets. Figure D.1 visualizes the results of the baseline methods with exact F_1 score.

D.3 Analysis of Emergent Languages

Emergent language hyperparameters

Due to the computational constraints for some of the environments, we stop CSAR after inducing the top 200 morphemes for each language as well as disabling the lookahead substitution heuristic. Additionally, for the CUB (natural images) environment, the following optimization were employed: limiting to the maximum meaning size considered to 3, pruning morpheme candidates that had a prevalence $\leq$1%, and consider only the 1000 most frequent forms and 1000 most frequent meanings.

The following hyperparameters were used for the vector observation environment:

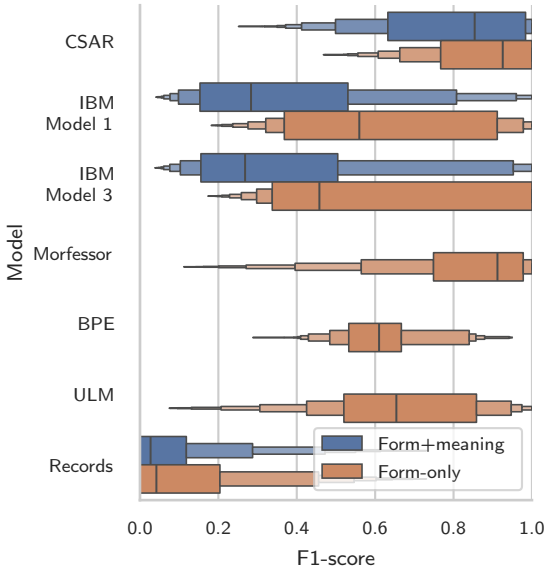


Figure D.1: Exact F_1 scores of baseline methods on the procedural datasets

***n* values** 4, 2 (sparse)
***n* attributes** 4, 8 (sparse)
***n* distractors** 3
vocab size 32
max sequence length 10
dataset size (CSAR input) 10 000 records

The ShapeWorld observation environment uses the following hyperparameters:

observations 5 shapes, 6 colors, 3 operators (and, or, not); *and* or *or* may only be used once
***n* examples** 20 total; 10 correct targets, 10 distractors
vocab size 30 (not including beginning- and end-of-sentence tokens)
max sequence length 8
dataset size (CSAR input) 1 200 records

The CUB observation environment uses the following hyperparameters (same as ShapeWorld unless noted):

observations 200 classes of birds with 40–60 images each; 312 binary attributes derived from the images
vocab size 18
***n* examples** 10 total
vision backbone ResNet-18

Both environments had any beginning-of-sentence and end-of-sentence tokens removed before being fed into CSAR.

	|Inv.|	|Form|
Vector, AV	94	3.59
Vector, sparse	126	3.51
SW, ref	2898	6.93
SW, setref	2920	8.13
SW, concept	1565	7.77

Table D.2: Metrics for form-only morpheme inventories generated by Morfessor across various emergent languages.

D.4 Morfessor Results on Emergent Language

In Table D.2 we show the results of running Morfessor on various emergent language corpora. Compared to the metrics for CSAR's output on the same corpora (Table 6.1), Morfessor's results do not match or even differ consistently (although Morfessor's forms do not have prevalence weighting like CSAR's). For the vector environments, Morfessor yields smaller inventories than CSAR yet larger inventories for ShapeWorld. Form lengths are similar for the vector environment, but for ShapeWorld, CSAR yields shorter forms than the vector environment while Morfessor yields much longer forms. Since we do not have ground truth morphemes for these emergent language corpora, we cannot definitively say one algorithm has performed better than the other. Yet Morfessor here is at a disadvantage here as it is not able to use the meanings of the utterances to guide its induction.

D.5 Morpheme Inventories

Top 100 morphemes induced by CSAR from human and emergent language datasets.

Human languages

Morpho Challenge (“’’, {+GEN}) (“ing$”, {+PCP1}) (“ed$”, {+PAST}) (“s”, {+PL}) (“er”, {er_s}) (“ly$”, {ly_s}) (“s$”, {+3SG}) (“ist”, {ist_s}) (“iz”, {ize_s}) (“ness”, {ness_s}) (“ion”, {ion_s}) (“ˆre”, {re_p}) (“ˆde”, {de_p}) (“ation”, {ation_s}) (“est$”, {+SUP}) (“ˆun”, {un_p}) (“less”, {less_s}) (“ful”, {ful_s}) (“ˆmis”, {mis_p}) (“head”, {head_N}) (“way”, {way_N}) (“ment”, {ment_s}) (“al”, {al_s}) (“it”, {ity_s}) (“ˆfire”, {fire_N}) (“ency$”, {ency_s}) (“hook”, {hook_N}) (“ish$”, {ish_s}) (“mind”, {mind_N}) (“ˆin”, {in_p}) (“at”, {ate_s}) (“if”, {ify_s}) (“able$”, {able_s}) (“ically$”, {ally_s}) (“ˆinter”, {inter_p}) (“ˆphoto”, {photo_p}) (“ˆhand”, {hand_N}) (“ˆscho”, {school_N}) (“house”, {house_N}) (“ical$”, {ical_s}) (“hold”, {hold_V}) (“long”, {long_A}) (“work”, {work_V}) (“up”, {up_B}) (“ag”, {age_s}) (“ant”, {ant_s}) (“ib”, {ible_s}) (“line”, {line_N}) (“ed$”, {ed_s}) (“er$”, {+CMP}) (“ˆover”, {over_p}) (“ˆdis”, {dis_p}) (“ˆsea”, {sea_N}) (“ˆim”, {im_p}) (“or”, {or_s}) (“pos”, {pose_V}) (“ence”, {ence_s}) (“ˆcardinal”, {cardinal_A}) (“ˆrational”, {rational_A}) (“ˆshoplift”, {shop_N}) (“conciliat”, {conciliate_V}) (“ˆmanicur”, {manicure_N}) (“ˆpredict”, {predict_V}) (“dressing”, {dressing_V}) (“ˆbuffet”, {buffet_V}) (“ˆcrimin”, {crime_N}) (“ˆentitl”, {entitle_V}) (“ˆfrivol”, {frivolous_A}) (“ˆheartb”, {heart_N}) (“ˆmaroon”,

{maroon_A}) (“^ribald”, {ribald_A}) (“^spread”, {spread_V}) (“^squeak”, {squeak_V}) (“^squint”, {squint_V}) (“^statue”, {statue_N}) (“^summar”, {summary_A}) (“whisper”, {whisper_V}) (“^blink”, {blink_V}) (“^carri”, {carry_V}) (“^cheer”, {cheer_V}) (“^four-”, {four_Q}) (“^hitch”, {hitch_V}) (“^louvr”, {louvre_N}) (“^muzzl”, {muzzle_N}) (“^nihil”, {nihilism_N}) (“^tooth”, {tooth_N}) (“^waist”, {waist_N}) (“guard$”, {guard_N}) (“^bull”, {bull_N}) (“^rail”, {rail_V}) (“^seri”, {series_N}) (“^test”, {test_N}) (“^two-”, {two_-Q}) (“ance$”, {ance_s}) (“board”, {board_N}) (“chain”, {chain_N}) (“eroom”, {room_N}) (“grand”, {grand_A}) (“order”, {order_V}) (“power”, {power_N})

Image captions (“tennis”, {tennis racket}) (“cat”, {cat}) (“train”, {train}) (“dog”, {dog}) (“pizza”, {pizza}) (“toilet”, {toilet}) (“man”, {person}) (“bus”, {bus}) (“clock”, {clock}) (“baseball”, {baseball glove}) (“frisbee”, {frisbee}) (“bed”, {bed}) (“horse”, {horse}) (“skateboard”, {skateboard}) (“laptop”, {laptop}) (“cake”, {cake}) (“giraffe”, {giraffe}) (“table”, {dining table}) (“bench”, {bench}) (“motorcycle”, {motorcycle}) (“bathroom”, {sink}) (“elephant”, {elephant}) (“umbrella”, {umbrella}) (“kitchen”, {oven}) (“kite”, {kite}) (“people”, {person}) (“ball”, {sports ball}) (“sheep”, {sheep}) (“zebra”, {zebra}) (“phone”, {cell phone}) (“surfboard”, {surfboard}) (“hydrant”, {fire hydrant}) (“zebras”, {zebra}) (“teddy”, {teddy bear}) (“truck”, {truck}) (“stop sign”, {stop sign}) (“sandwich”, {sandwich}) (“boat”, {boat}) (“street”, {car}) (“bat”, {baseball bat}) (“bananas”, {banana}) (“giraffes”, {giraffe}) (“living”, {couch}) (“snow”, {skis}) (“bird”, {bird}) (“elephants”, {elephant}) (“vase”, {vase}) (“cows”, {cow}) (“broccoli”, {broccoli}) (“computer”, {keyboard}) (“woman”, {person}) (“tie”, {tie}) (“horses”, {horse}) (“bear”, {bear}) (“desk”, {mouse}) (“plane”, {airplane}) (“luggage”, {suitcase}) (“airplane”, {airplane}) (“person”, {person}) (“hot”, {hot dog}) (“refrigerator”, {refrigerator}) (“wii”, {remote}) (“kites”, {kite}) (“boats”, {boat}) (“couch”, {couch}) (“traffic”, {traffic light}) (“plate”, {fork}) (“surf”, {surfboard}) (“umbrellas”, {umbrella}) (“wine”, {wine glass}) (“skate”, {skateboard}) (“bowl”, {bowl}) (“stuffed”, {teddy bear}) (“room”, {tv}) (“cow”, {cow}) (“scissors”, {scissors}) (“snowboard”, {snowboard}) (“chair”, {chair}) (“car”, {car}) (“banana”, {banana}) (“bicycle”, {bicycle}) (“birds”, {bird}) (“vegetables”, {broccoli}) (“microwave”, {microwave}) (“donuts”, {donut}) (“video”, {remote}) (“batter”, {baseball bat, person}) (“skateboarder”, {person, skateboard}) (“surfer”, {person, surfboard}) (“skis”, {skis}) (“motorcycles”, {motorcycle}) (“meter”, {parking meter}) (“suitcase”, {suitcase}) (“sink”, {sink}) (“bike”, {bicycle}) (“chairs”, {chair}) (“food”, {bowl}) (“dogs”, {dog}) (“oven”, {oven}) (“court”, {sports ball})

Machine translation (“and”, {und}) (“Commission”, {Kommission}) (“not”, {nicht}) (“Union”, {Union}) (“we”, {wir}) (“I”, {ich}) (“that”, {daß}) (“Mr”, {Herr}) (“I”, {Ich}) (“Parliament”, {Parlament}) (“President”, {Präsident}) (“Member States”, {Mitgliedstaaten}) (“report”, {Bericht}) (“European”, {Europäischen}) (“We”, {Wir}) (“or”, {oder}) (“in”, {in}) (“Europe”, {Europa}) (“the”, {der}) (“Council”, {Rat}) (“between”, {zwischen}) (“is”, {ist}) (“2000”, {2000}) (“Commissioner”, {Kommissar}) (“EU”, {EU}) (“for”, {für}) (“the”, {die}) (“The”, {Die}) (“also”, {auch}) (“with”, {mit}) (“like to”, {möchte}) (“you”, {Sie}) (“1999”, {1999}) (“directive”, {Richtlinie}) (“only”, {nur}) (“proposal”, {Vorschlag}) (“European”, {Europäische}) (“Madam”, {Präsidentin}) (“Mrs”, {Frau}) (“Kosovo”, {Kosovo}) (“but”, {aber}) (“new”, {neuen}) (“Group”, {Fraktion}) (“have”, {haben}) (“behalf”, {Namen}) (“Mr”, {Herrn}) (“women”, {Frauen}) (“has”, {hat}) (“regions”, {Regionen}) (“years”, {Jahren}) (“all”, {alle})

(“two”, {zwei}) (“cooperation”, {Zusammenarbeit}) (“if”, {wenn}) (“1”, {1}) (“new”, {neue}) (“Article”, {Artikel}) (“because”, {weil}) (“whether”, {ob}) (“Parliament”, {Parlaments}) (“a”, {eine}) (“measures”, {Maßnahmen}) (“but”, {sondern}) (“institutions”, {Institutionen}) (“social”, {sozialen}) (“to”, {zu}) (“political”, {politischen}) (“development”, {Entwicklung}) (“national”, {nationalen}) (“today”, {heute}) (“countries”, {Länder}) (“European”, {europäischen}) (“must”, {muß}) (“our”, {unsere}) (“as”, {wie}) (“problems”, {Probleme}) (“initiative”, {Initiative}) (“work”, {Arbeit}) (“be”, {werden}) (“very”, {sehr}) (“human rights”, {Menschenrechte}) (“of the”, {des}) (“us”, {uns}) (“three”, {drei}) (“debate”, {Aussprache}) (“other”, {anderen}) (“hope”, {hoffe}) (“already”, {bereits}) (“question”, {Frage}) (“this”, {diesem}) (“debate”, {Debatte}) (“are”, {sind}) (“will”, {wird}) (“proposals”, {Vorschläge}) (“If”, {Wenn}) (“Prodi”, {Prodi}) (“Council”, {Rates}) (“rapporteur”, {Berichterstatter}) (“INTERREG”, {INTERREG}) (“role”, {Rolle})

Emergent languages

Below are the top 20 morphemes for the various emergent languages described in Section 6.5. For attr–val environments, a semantic component of “1_2” means the 1st attribute has a value of 2.

Vector, Attr–val, Discrim, Run 0 (“17”, {3_3}), (“16”, {1_3}), (“22”, {0_1}), (“24”, {3_1}), (“40”, {2_1}), (“22 39”, {0_0}), (“22”, {1_0}), (“46 44 46 46 44 46 46”, {0_1, 1_0, 3_2}), (“46”, {0_1}), (“31 31 31 31”, {3_2}), (“46”, {1_0}), (“22 5 11 5”, {0_0, 2_2}), (“24 24”, {2_1}), (“24 44”, {3_0}), (“11”, {2_2}), (“11”, {2_3}), (“5”, {3_1}), (“39 39”, {0_0, 1_0}), (“24 22 24”, {2_1}), (“44 31”, {0_2, 1_1})

Vector, Attr–val, Discrim, Run 1 (“7”, {0_2}), (“58 58”, {0_1}), (“4 4 4”, {0_0}), (“52”, {2_1}), (“4 0”, {3_3}), (“4 58 4”, {0_3}), (“19 19 40”, {1_3, 2_2}), (“19 19 40 40”, {1_2, 2_2}), (“21 21 21 21 21 21 21 0”, {3_0}), (“60”, {2_3}), (“19 40 40 40”, {1_2, 2_0}), (“40 40 40 40 40”, {1_2, 2_3}), (“21 21”, {3_1}), (“19 40 40”, {1_3, 2_0}), (“19 19 19”, {1_2}), (“21 21 21”, {3_0}), (“19 40 4”, {1_1, 2_0}), (“21”, {3_2}), (“48 40”, {1_1, 2_3}), (“58 21 58”, {0_1})

Vector, Attr–val, Discrim, Run 2 (“10”, {1_1}), (“58 58 58 58 58 0”, {2_1}), (“24 24 24 24 24 0”, {2_3}), (“63 26”, {1_3}), (“53”, {1_0}), (“35”, {3_0}), (“46”, {1_2}), (“45 45”, {0_3}), (“56”, {2_0}), (“58”, {2_2}), (“24 0”, {2_0}), (“26 26 26”, {1_3}), (“45 3 3”, {1_2, 3_3}), (“45 63 63”, {3_2}), (“45 26 3”, {1_0, 3_3}), (“45”, {0_2, 3_1}), (“3 3 24 24”, {3_3}), (“24 24 24”, {3_3}), (“63 63 63 63”, {3_0}), (“53 63 53 63”, {0_0, 3_0})

Vector, Attr–val, Recon, Run 0 (“12 12 12”, {2_2}), (“29 29 29”, {1_2}), (“40 40 40”, {2_1}), (“11”, {1_1}), (“13”, {0_2}), (“14”, {0_1}), (“16”, {3_1}), (“17”, {2_0}), (“31”, {3_0}), (“37”, {1_0}), (“4”, {0_3}), (“41”, {3_2}), (“44”, {2_3}), (“59”, {3_3}), (“8”, {0_0}), (“50”, {1_3}), (“41 8 41 8 41 8”, {1_3}), (“17 16”, {1_3}), (“4 41 4 41 4 41 4 41”, {1_3}), (“41 13 41 13 17 41 17”, {1_3})

Vector, Attr–val, Recon, Run 1 (“21 21”, {0_2}), (“10”, {2_1}), (“13”, {1_1}), (“14”, {3_1}), (“37”, {0_0}), (“42”, {2_2}), (“46”, {3_0}), (“47”, {0_1}), (“48”, {3_3}), (“6”, {1_3}), (“61”, {0_3}), (“16”, {2_3}), (“38”, {1_2}), (“17”, {2_0}), (“17”, {3_2}), (“17”, {1_0}), (“28 28 28 28 0”, {1_2}), (“28”, {2_3})

Vector, Attr–val, Recon, Run 2 (“27 27 0”, {0_0}), (“63 63 0”, {0_2}), (“4 4”, {3_0}), (“61 61”, {3_2}), (“11”, {3_1}), (“14”, {1_1}), (“47”, {1_0}), (“6”, {1_2}), (“56”, {1_3}), (“1”, {2_1}), (“26”, {2_0}), (“21”, {2_3}), (“18 0”, {0_3}), (“22 22 22 22 0”, {0_3, 2_2}), (“31 31 31 31 0”, {0_1, 2_3}), (“10 10 10 0”, {0_1, 2_2}), (“29”, {2_2}), (“40 40 40 40”, {0_1, 2_0}), (“55 55”, {0_1, 2_1}), (“55”, {2_1})

Vector, Sparse, Discrim, Run 0 (“5 5”, {1, 5}), (“11”, {1, 2}), (“52”, {7}), (“2”, {2, 5}), (“5”, {1}), (“6”, {2}), (“16”, {2, 4}), (“20”, {5}), (“34”, {3, 4}), (“34 34 34”, {4}), (“34 34”, {4}), (“52 38”, {0, 1, 4}), (“52 52 52”, {5}), (“34”, {4}), (“5”, {5}), (“2”, {2}), (“26 45 26”, {5, 7}), (“11”, {1}), (“11”, {2}), (“38 38”, {0})

Vector, Sparse, Discrim, Run 1 (“50”, {2}), (“16”, {4}), (“24”, {2, 7}), (“25”, {1}), (“42”, {0}), (“54”, {7}), (“24”, {2}), (“45 45”, {7}), (“36 36”, {0}), (“24”, {7}), (“36”, {5}), (“42”, {2}), (“34 34 42”, {1, 4, 5}), (“18”, {1, 5}), (“25 25 25 25 25”, {4}), (“42 42 42”, {5}), (“10 34”, {1, 4}), (“36”, {0}), (“52 52”, {1, 7}), (“52 9 52 9 52”, {0, 4, 5})

Vector, Sparse, Discrim, Run 2 (“9 0”, {2, 5}), (“34”, {5}), (“32 32 32”, {1, 7}), (“7”, {4}), (“33 33”, {7}), (“1”, {1}), (“5 5”, {2}), (“9”, {5}), (“55 0”, {1, 2, 6}), (“32”, {3, 6}), (“55 55”, {0, 5}), (“36”, {1}), (“49 49 48”, {0, 3, 4, 6}), (“9 9 9 9”, {2}), (“2 2 2”, {0, 4}), (“48 48 48 48 48”, {2, 4}), (“33 32”, {7}), (“33”, {3}), (“14 5”, {6}), (“55 55”, {5, 6})

Vector, Sparse, Recon, Run 0 (“2”, {6}), (“53”, {5}), (“27”, {0}), (“28”, {1, 2}), (“28”, {1}), (“26”, {1, 5}), (“26”, {5}), (“28”, {2, 3}), (“22 8”, {2, 5}), (“49 53”, {2, 3}), (“12 12 58”, {0}), (“58 5 58”, {2}), (“50”, {4}), (“5 49 5”, {2, 4}), (“5 5”, {2, 4}), (“36”, {7}), (“8 53 8”, {1, 2}), (“38 5 38”, {4}), (“49 58”, {3}), (“8 26”, {1})

Vector, Sparse, Recon, Run 1 (“8”, {4, 5}), (“50”, {1, 2}), (“21”, {3, 7}), (“42”, {1}), (“50”, {1}), (“31 31”, {5}), (“19”, {4}), (“33”, {2}), (“1”, {0}), (“23”, {3, 6}), (“62”, {7}), (“50”, {2}), (“37”, {5, 6}), (“23”, {7}), (“37”, {4, 6}), (“47”, {7}), (“23”, {6}), (“8”, {4}), (“21”, {3, 5}), (“47”, {3})

Vector, Sparse, Recon, Run 2 (“19 19”, {2, 6}), (“4”, {4, 7}), (“58”, {3}), (“12”, {4}), (“14”, {7}), (“10 10”, {6}), (“34”, {2}), (“56 56”, {0, 1, 3}), (“56 23”, {0, 1, 5}), (“4”, {4}), (“23”, {0, 5}), (“4”, {7}), (“56”, {1, 3}), (“59 59”, {3, 5}), (“30”, {5}), (“3”, {3}), (“23”, {1, 5}), (“51”, {1}), (“56 58”, {0, 7}), (“8 8”, {0, 3})

Image, Synth, Ref, Run 0 (“6”, {gray}), (“3”, {square}), (“4”, {rectangle}), (“6 15 6”, {circle, yellow}), (“4 8”, {square}), (“3 8 8 4”, {circle, red}), (“5 6 15 5 6”, {blue, triangle}), (“14 10 14 10”, {triangle}), (“16 8 6”, {green, square}), (“4 10 10 4”, {circle, red}), (“8 6 10”, {ellipse}), (“6 12 5”, {yellow}), (“8 4 13”, {square, white}), (“4 4 4”, {green}), (“12 12 12 6 12”, {circle, green}), (“4 6”, {square}), (“5 14 5”, {red}), (“5 5 15 14”, {ellipse, red}), (“4 8 8 4”, {triangle, yellow}), (“15”, {rectangle})

Image, Synth, Ref, Run 1 (“8 11”, {square}), (“16 4 16”, {square}), (“16 4”, {square}), (“4 16”, {square}), (“5 16”, {square}), (“16 4”, {triangle}), (“13 0 11”, {gray, rectangle}), (“16 0 16”, {white}), (“0 16”, {ellipse}), (“0 8 8”, {green}), (“4 7 4 4 7”, {ellipse, gray}), (“11 11 4 16 11”, {ellipse, gray}), (“16 11 4 4”, {green, triangle}), (“8 16 16”, {ellipse}), (“4 8 8 8”, {green, rectangle}), (“5 8 7 5 7”, {red, square}), (“14”, {square}), (“0 0 11 16”, {ellipse}), (“7 7 4 4 7”, {red, triangle}), (“13 14 16 16”, {triangle, yellow})

Image, Synth, Ref, Run 2 (“7”, {rectangle}), (“4”, {gray}), (“15 15 9”, {square}), (“9”, {square}), (“9”, {ellipse}), (“9”, {gray}), (“9”, {rectangle}), (“7 15 15”, {red, square}), (“13 8 8”, {red, square}), (“4 7 7 15”, {blue, circle}), (“14 14 13 7”, {circle, green}), (“7 7 7 15 7”, {square, yellow}), (“15 15 15 9”, {blue, circle}), (“0 0 0 8 0”, {white}), (“0 0 0 4 8”, {red, square}), (“13 15 13”, {green}), (“15 4 4 4 15”, {blue, triangle}), (“0 0 13 0 0”, {rectangle, yellow}), (“13 15 0”, {red}), (“0 0 7 4”, {green, triangle})

Image, Synth, Set-ref, Run 0 (“13”, {circle, not}), (“11 11”, {blue, not}), (“12 12”, {ellipse}), (“3 3 3”, {gray, not}), (“10”, {not, or}), (“12”, {not, rectangle}), (“8 11 7”, {green, or, yellow}), (“7”, {not, red}), (“11 8 8”, {or, red, yellow}), (“9”, {triangle}), (“14 8”, {gray, or, yellow}), (“8”, {and, not, white}), (“16 7 7”, {blue, or, white}), (“14 14 7”, {or, white, yellow}), (“11 3 3”, {green, or, red}), (“16 14 14”, {gray, or, white}), (“7 8”, {blue, or, yellow}), (“14 14 11”, {and, blue, green}), (“10”, {not, triangle}), (“3 3 8”, {blue, or, red})

Image, Synth, Set-ref, Run 1 (“9”, {or, yellow}), (“10”, {not, yellow}), (“3 3”, {gray, not}), (“14 14”, {blue, not}), (“7”, {and, not, white}), (“5”, {not, red}), (“12”, {not, rectangle}), (“11”, {or, white}), (“12 12”, {ellipse}), (“10”, {red}), (“4 4”, {triangle}), (“5 5 5”, {blue}), (“11 7”, {gray}), (“12”, {ellipse, or}), (“4”, {or, triangle}), (“7 7”, {green, not, white}), (“5”, {blue, or}), (“4”, {circle, not}), (“9 9”, {yellow}), (“10 11”, {and, green, yellow})

Image, Synth, Set-ref, Run 2 (“12”, {not, red}), (“8 8”, {blue}), (“10 10”, {not, yellow}), (“0 0”, {blue, not}), (“15”, {or, red}), (“5 5”, {triangle}), (“3”, {not, rectangle}), (“13”, {circle, not}), (“15”, {gray, not}), (“15”, {and, not, white}), (“3 10”, {circle, ellipse, or}), (“3 3 3”, {ellipse}), (“9 9”, {white}), (“13”, {or, yellow}), (“3”, {ellipse, or}), (“12 12”, {red}), (“9 10”, {ellipse, not, triangle}), (“5”, {square}), (“10 12 9”, {gray, or, white}), (“0 10”, {and, blue, yellow})

Image, Synth, Concept, Run 0 (“13 14”, {not, yellow}), (“15 13”, {not, red}), (“16”, {yellow}), (“15 15”, {gray, not}), (“14 3”, {red}), (“16”, {rectangle}), (“3”, {blue, not}), (“16”, {circle}), (“4 4”, {white}), (“5 5 5”, {and, not, white}), (“9 9 9”, {green}), (“5 15 15”, {blue,

or, yellow}), (“16”, {triangle}), (“16”, {ellipse}), (“5 3 3”, {or, red, yellow}), (“15 3”, {yellow}), (“13 13”, {not, yellow}), (“3 13 13”, {gray, or, white}), (“3 5”, {gray, or, yellow}), (“3 15 15”, {or, white, yellow})

Image, Synth, Concept, Run 1 (“11 11”, {blue}), (“4”, {not, yellow}), (“16 16”, {not, red}), (“16 11”, {gray, not}), (“8”, {or, yellow}), (“14 14”, {blue, not}), (“8”, {not, white}), (“14 4”, {white}), (“3 3”, {green, or, red}), (“11 8”, {and, green, white}), (“8 8 8”, {yellow}), (“11 3”, {and, white, yellow}), (“3 3”, {and, blue, white}), (“13”, {red}), (“5 5”, {not, or}), (“14 14 16”, {white, yellow}), (“10 10 10 10”, {and, green}), (“4”, {or, white}), (“3”, {blue, not}), (“4”, {and, yellow})

Image, Synth, Concept, Run 2 (“8 8”, {not, yellow}), (“6”, {not, red}), (“0 0”, {blue, not}), (“0 11”, {gray}), (“6 6”, {gray, not}), (“15”, {yellow}), (“15”, {rectangle}), (“3 0”, {or, white, yellow}), (“0”, {white}), (“11 6 6”, {blue, or, yellow}), (“11 11 8”, {blue, or, white}), (“15”, {circle}), (“11”, {and, green, not}), (“15”, {triangle}), (“15”, {ellipse}), (“8 6”, {red}), (“8 3 3”, {green, not, or}), (“11 11”, {blue}), (“11 0”, {gray, or, yellow}), (“15”, {square})

Image, Natural, Ref, Run 0 (“22”, {221}), (“22”, {226}), (“22”, {46}), (“7”, {204, 46}), (“22”, {255}), (“22”, {236}), (“12”, {146, 152, 245}), (“18”, {146, 236}), (“6 22”, {102, 165, 195}), (“22”, {102, 228}), (“22 22”, {152}), (“6”, {46}), (“22”, {6}), (“6”, {204}), (“22”, {150, 306}), (“7”, {236}), (“12”, {127, 55}), (“3”, {152, 245, 8}), (“22”, {146, 152, 55}), (“20”, {146, 221})

Image, Natural, Ref, Run 1 (“6”, {221}), (“6”, {226}), (“11”, {46}), (“11”, {236}), (“11”, {22, 306}), (“6”, {236}), (“11”, {6}), (“11”, {226}), (“10”, {236}), (“6”, {146, 152}), (“6”, {146}), (“22”, {146, 255}), (“11”, {222}), (“6”, {152}), (“11”, {150, 30, 306}), (“7”, {5}), (“20”, {211, 262, 76}), (“18”, {204}), (“6”, {219}), (“11”, {146, 7})

Image, Natural, Ref, Run 2 (“17”, {221}), (“7”, {221}), (“17”, {236}), (“0”, {5}), (“18”, {221}), (“17”, {204}), (“7”, {226}), (“0”, {165}), (“17”, {146, 152}), (“0”, {146, 222}), (“18 18”, {102, 92}), (“0”, {195, 306}), (“16 3”, {102, 210, 92}), (“0”, {146, 245, 8}), (“10”, {6}), (“18”, {255}), (“18”, {261, 52, 92}), (“7”, {148}), (“18”, {117, 164, 36}), (“16”, {222})

Image, Natural, Set-ref, Run 0 (“7 7 7 7 7 7 7 7”, {204, 236, 255}), (“6 6”, {241, 260, 91}), (“5”, {221}), (“10”, {5}), (“5”, {226}), (“10”, {146, 195, 245}), (“15 15 15”, {249, 294}), (“12”, {219}), (“6”, {245, 260, 91}), (“10”, {30, 306}), (“3 3”, {310}), (“15”, {210, 7}), (“10”, {221, 245}), (“21”, {150, 195, 55}), (“3 3”, {11, 60}), (“14”, {166}), (“7”, {150, 37}), (“7”, {204, 255}), (“7 7”, {236, 241}), (“7”, {236, 255, 46})

Image, Natural, Set-ref, Run 1 (“6”, {236}), (“17 3”, {255}), (“6”, {221}), (“4 4”, {307}), (“18 18”, {165, 55}), (“13”, {241, 245, 260}), (“17”, {46}), (“6 6”, {226}), (“12 18”, {245, 306}), (“17”, {146, 236}), (“4 4”, {166, 219}), (“19”, {7}), (“5”, {310}), (“15”, {55}), (“19 19”, {153, 80}), (“18”, {146, 210}), (“6”, {152, 219}), (“17”, {204}), (“6”, {219}), (“19”, {236, 245, 52})

Image, Natural, Set-ref, Run 2 (“18”, {236}), (“17”, {255}), (“18 21”, {226}), (“18”, {221}), (“14 14”, {245, 260, 275}), (“18”, {219}), (“0 18”, {165, 55}), (“14 14”, {275, 55, 70}), (“18”, {7}), (“18”, {146, 152}), (“0 0”, {245, 306}), (“10 10”, {307}), (“17”, {146, 236}), (“16 16”, {10, 153}), (“0”, {260}), (“16”, {7}), (“18”, {152}), (“0”, {36, 70}), (“0”, {146, 245, 30}), (“22”, {112, 236})

Image, Natural, Concept, Run 0 (“14 17”, {255}), (“15 15”, {245, 306}), (“9”, {219}), (“9”, {221}), (“15”, {146, 195, 245}), (“20”, {305, 309, 36}), (“15 15”, {165, 55}), (“17”, {46}), (“20”, {164, 179, 241}), (“10”, {166}), (“15”, {150, 306}), (“8 8”, {117, 152, 55}), (“10”, {307}), (“15”, {146, 195, 55}), (“8”, {132, 209, 219}), (“8”, {132, 219, 51}), (“8”, {102, 132, 275}), (“16”, {118, 236, 241}), (“20”, {194, 309}), (“17”, {204})

Image, Natural, Concept, Run 1 (“19 19 19 19”, {226}), (“12”, {255}), (“11 11 11 11 11”, {245, 306}), (“11 11”, {146, 195, 55}), (“6 6”, {260, 290, 55}), (“12”, {46}), (“11”, {5}), (“6 6 6”, {290, 55, 70}), (“6 6”, {194, 209, 219}), (“15”, {204}), (“12”, {112}), (“19”, {221}), (“0”, {146, 236, 7}), (“6”, {219, 245, 51}), (“16”, {249, 294}), (“15”, {146, 236}), (“16”, {10, 152}), (“19”, {219}), (“20”, {118, 236, 241}), (“4”, {210, 236})

Image, Natural, Concept, Run 2 (“11 11 11 11 11 11 11 11”, {221}), (“8 5”, {112, 255}), (“8”, {255, 46}), (“5”, {112}), (“8”, {255}), (“5”, {204}), (“8”, {46}), (“16 16 16”, {152, 249}), (“15”, {53}), (“21”, {210, 236, 52}), (“16 16”, {249, 294}), (“11 11 11”, {179, 51}), (“17”, {219, 241, 52}), (“16”, {10, 152}), (“15”, {119}), (“9”, {36}), (“11 11”, {21, 51}), (“9”, {179}), (“9”, {21}), (“9”, {222})