The Emergence of Compositional Languages in Multi-entity Referential Games: from Image to Graph Representations

Daniel Akkerman
Department of Cognitive Science and Artificial Intelligence.
Tilburg University
d.akkerman.d@gmail.com

Phong Le
CKHub
lephong.xyz@gmail.com

Raquel G. Alhama
Institute for Logic, Language and Computation.
University of Amsterdam
rgalhama@uva.nl

Abstract

To study the requirements needed for a human-like language to develop, Language Emergence research uses jointly trained artificial agents which communicate to solve a task, the most popular of which is a referential game. The targets that agents refer to typically involve a single entity, which limits their ecological validity and the complexity of the emergent languages. Here, we present a simple multi-entity game in which targets include multiple entities that are spatially related. We ask whether agents dealing with multi-entity targets benefit from the use of graph representations, and explore four different graph schemes. Our game requires more sophisticated analyses to capture the extent to which the emergent languages are compositional, and crucially, what the decomposed features are. We find that emergent languages from our setup exhibit a considerable degree of compositionality, but not over all features.[1]

1 Introduction

In Language Emergence research, jointly trained artificial agents develop their own communication protocol to solve a task, the most popular of which is the *referential* game – a version of the classic *signaling game* (Lewis, 1969; Skyrms, 2010). In such games, a *sender* agent composes a message to communicate about a target, and a *receiver* agent tries to identify which target was the sender referring to, by selecting it from a set of items (e.g. Lazaridou et al., 2017; Havrylov and Titov, 2017; Chaabouni et al., 2022).

This setting has shown to be a successful environment for a communication protocol to emerge from scratch (Lazaridou and Baroni, 2020). Agents learn to successfully communicate about target items which represent entities, e.g. chairs, bicycles, dogs, or cats. However, while accurate communication appears to be achieved with relative ease in these systems, the emerged languages miss some of the interesting properties of natural language. In particular, human languages exhibit a strong bias towards *compositionality*, a type of systematicity according to which parts of a message (for example morphemes, words, or phrases) systematically refer to parts of the input (Partee, 2008). Crucially, choices in the representation of input items have an influence on the emergence of compositionality, such that image-based setups have been less successful than settings that use manually coded feature-based vectors (Lazaridou et al., 2018).

Notwithstanding the current successes of the systems described above, a relevant limitation of such games is that they are trained to communicate about single entities e.g., one cat or one bicycle) or higher-level descriptions of single entities (or *concepts*, Mu and Goodman, 2021). While this simplification has proved to be useful to investigate basic requirements for emergence of a communication protocol, clearly single entities alone are not sufficient to represent world knowledge, let alone support communication about any state of affairs that goes beyond entity naming.

However, introducing multiple entities in a referential game is not trivial, in particular when it comes to input representations. To the best of our knowledge, only Lian et al. (2023) addressed this challenge using items involving two entities (agent and patient), but their agents require supervised training over a hand-crafted miniature language. One reason behind the complexity of a multientity game comes from the input representations: although image-based input would be naturally suited to seemingly represent multiple entities, compositionality does not seem to emerge easily in such systems. On the other hand, feature-based vectors do exhibit more tendency to encourage compositional languages, but they do not scale gracefully to represent multiple entities.

[1]Our implementation is available at https://github.com/Lumalizer/Gridgame_Emnlp2024.git.

To tackle this challenge, we propose the use of graphs to represent inputs. Graphs are naturally suited for encoding relationships between multiple entities, which can coexist within the same graph. Like feature-based vectors, they have the advantage of being disentangled (i.e. the structure of the input is explicit), hence *a priori* they have potential to encourage the emergence of compositionality. And like images, neural graph embeddings scale gracefully, such that graphs of different sizes can be seamlessly integrated within a vector of fixed size. Since there is no standard way to encode information in a graph, we propose four different schemas and analyze their relative contributions to accuracy of communication and compositionality.

To perform our experiments, we present a new, simple game in which two entities coexist in a 4-positional grid. These entities are *spatially* related, such that one may be, for example, above or to the left of the other. We choose spatial relations because they can also be visually represented. Thanks to this, our game allows us to compare graph representations, which have a higher level of abstraction, to image-based input, which is a lower-level representation. Loosely speaking, these representations can be interpreted as conceptual (in the case of graphs) and perceptual (in the case of images).

Albeit simple, our representations incorporate an additional level of complexity by introducing multiple entities, as well as the relation between them. This entails that the analyses of the emergent language also comes with new challenges, since there are multiple ways in which a language could exhibit a degree of compositionality –depending on which features of the input are decomposed. To address this challenge, we present a methodology for tracking compositionality that builds on previously used metrics, but applies an extension that allows for more fine-grained analyses.

Our work combines these three ingredients (a new multi-entity game, a variety of input representations, and an extended analysis method) to explore the requirements for communication success and compositionality to emerge in a scenario involving multiple entities.

2 Background

In this section we briefly introduce referential games and agent architectures, with special focus on input representations – since that is particularly relevant to our work.

2.1 Referential games

A referential game (Lewis, 1969; Steels, 2015) involves two agents (sender and receiver) communicating to cooperatively select specific items. The sender observes a target item and generates a message to describe it. The receiver interprets the message and selects the target among several other items (called distractors). If the receiver consistently selects targets, we say that their communication is successful, therefore they agreed on a communication protocol; in other words, a language emerged from their interaction. Items in such games can be anything, but are often chosen to encourage language emergence. The most popular type of item represents an entity (e.g., an object such as a bicycle) characterized by distinct features, such as color, shape, and size (Lazaridou et al., 2017; Ren et al., 2020; Chaabouni et al., 2021).

Agents access items via an input representation (e.g. pixels or descriptive attributes). The sender transforms the input representation of the target item into an embedding and then generates one or more tokens, using e.g. a recurrent network (Havrylov and Titov, 2017). This sequence of tokens is called a message. The receiver will then encode the message into an embedding and compare that embedding against the embeddings of a given set of items. The item with the most similar embedding will be selected. The two agents are trained to maximize the times the target item is selected, or *communication success*.

If communication success is achieved, we analyze the emergent language to see if it has similar properties to human language. *Compositionality* is seen as the hallmark property of our communication system, as it allows us to construct unlimited expressions with a finite vocabulary. To illustrate this concept with a simple example: in a non-compositional language, there would be a different, unrelated word for each entity that had a different property; for instance, a yellow bicycle could be named “gug” and a brown bicycle could be called “perflor”. There is nothing in common between those two words that indicates that they refer to the same type of entity; in addition, we would need to come up with a new word when encountering a bicycle of another color. Instead, in a compositional language, we would separate entity type from color, such that expressions like “yellow bicycle” and “brown bicycle” consist of two parts that systematically refer to color and entity. In this way, words for color and entity can be seamlessly combined to refer to other combinations of entities and color, such as “pink bicycle” or “yellow car”.

2.2 Input representations

We introduce the two types of representations that neural approaches to language emergence have mostly focused on: feature vectors (which embed properties of a represented entity) and images. We then present graph representations, which is the type of input we focus on in this paper.

Feature vectors. One way to construct a feature vector is by giving a specific meaning to every dimension of the vector, such that each dimension represents a property. For instance, for a vector representing a *tiger*, dimensions may refer to features such as *has_whiskers* and *is_striped* which would have a value of 1, while other dimensions may refer to features such as *has_wheels* which would have a value of 0. An alternative method to construct feature vectors is to concatenate one-hot vectors for each specific feature (e.g. Kottur et al., 2017; Chaabouni et al., 2020).

In both representations the meaning of each vector dimension remains constant across items, therefore keeping features disentangled. The use of disentangled vectors has the advantage of transparently providing the system with clearly separated features, hence relieving the agents form the task of discovering such structure (Lazaridou et al., 2018). However, a disadvantage of this representation is that it has a fixed size, so it lacks the flexibility of representing a varying number of entities.

Images. Another line of work introduced the use of images to represent the input (Havrylov and Titov, 2017; Lazaridou et al., 2017; Evtimova et al., 2018; Bouchacourt and Baroni, 2018). One argument that is often put forward in support of these representations is their ecological validity: pixels can be thought of as a representation of lower-level visual perception, hence researchers do not need to make assumptions regarding which conceptual features play a role in communication (as is the case when using disentangled vectors).

When using images, the properties of the represented entities are *entangled*: they are not clearly separated, but rather spread over pixels, the location and value of which may differ from image to image. For instance, following the previous example, whiskers may appear in different locations in different images. Thus, when using this type of representation, the agents have the additional task of discovering relevant structure in the input (which may entail entity segmentation and categorization). Thus, perhaps non-surprisingly, systems using images seem to be less inclined to develop languages that are compositional (Lazaridou et al., 2018). However, a relevant advantage of this type of representation is that multiple entities can be seamlessly represented within the same image, without altering vector size.

Graphs. To the best of our knowledge, language emergence with graph-based input has only been investigated in Słowik et al. (2020); Slowik et al. (2020). The authors use two graph setups. The first one uses randomly generated graphs —and is therefore less relevant to our work, as it does not describe any meaningful input. The second uses tree-like graphs to describe entities, such that the root node represents the entity and the leaves represent features of the depicted entity.

We postulate that graphs keep the best properties of feature vectors and images. They offer a disentangled description of the input, which may support the emergence of compositional languages (as is the case in the experiments reported in the above-cited work). However, unlike feature-based vectors, they *scale gracefully*, such that multiple entities can be represented seamlessly within the same graph, thanks to the fact graph representations can handle an arbitrary number of nodes, node features, edges, and edge features. Importantly, these can be embedded in vectors of fixed size, even for large graphs of e.g. thousands of nodes and edges (Wu et al., 2019). Although the same is true for image-based representations (i.e. image size is fixed regardless of image content), this is not the case for feature-based vectors representing multiple entities.

3 Multi-entity Game

We postulate that language emergence setups need to eventually account for games that represent more than one entity, such that more complex languages can potentially emerge –after all, some of the most common constructions in human language use involve a subject and (at least) one object. However, extending current referential games to handle multi-entity input is not trivial, as we first need to explore which type of input representations support multi-entity representation while ensuring communication success and favoring compositionality. We now present our proposal of a simple multi-entity referential game and describe the input representations we experiment with.

3.1 Game Design

Conceptually, our game is similar to referential single-entity games. The key difference is that an item in our game represents a collection of entities placed in a specific relationship. To make the game accessible to vision-based agents, we choose this relationship to be *spatial*. Hence, in our game, an item consists of a 4-positional grid, in which each position can host one entity –depicted as a simple shape, such as an eagle or a rabbit (see Figure 1 for an example). It is worth noting that, for simplicity, here an entity has only a shape feature. It is straightforward to add more features (e.g. color, size) to entities.

Thus, one advantage of our game is that this class of items is easy to represent with an image, but also with a graph. Additionally, these items can be described using a simple script-like description (e.g. "eagle top left rabbit bottom right"), which is particularly useful to analyse the emergent languages (as we explain later in section 6).

3.2 Graph Representations

We now describe the graph representations that we use in our game. First we introduce a distinction between the three types of nodes that we use. An **entity node** represents an entity, together with its properties. The properties can take any shape; for instance, a feature-based vector. Here, for simplicity, we use a one-hot encoding with the index of the shape of the entity (e.g. "eagle"). A **position node** represents a position, which can be the absolute position of an entity in the grid (e.g. "top-left") or its relative position with respect to another entity. To represent the latter, we subtract the absolute positions of each entity (see Figure 1). An **entity-position node** encodes the position of the entity as a feature, and thus the attribute vector is a concatenation of the one-hot vector representing the shape and the vector representing the absolute position of the entity in the grid.

We use these node types to construct four types of graphs, illustrated in figure 1. The first two types use absolute positions, which are encoded as attributes of their corresponding entity nodes. The *graph-posattr*, or "position as node attribute" representation contains two entity-position nodes connected via an undirected edge. In *graph-leaves*, or "position as leaf", two position-nodes are connected to their corresponding entity-nodes with a directed edge.

The other two types use relative positions. In *graph-edge*, or "position as edge label", two entity nodes are connected via two directed edges labeled with their relative positions. Finally, a relative position can be formulated as a functor and its entities as functees, resulting in a logical expression like `above-left(eagle, rabbit)`. We call this *graph-functor*, or "position as functor".

3.3 Model

An additional advantage of our new game is that existing agent architectures for single-entity games (described in Section 2.1) can be reused.

Let's use $\mathcal{V}$ for the vocabulary and $\mathcal{I}$ for a set of items. In our game, there are two neural network agents called sender and receiver. The sender agent A_{S_θ}, a neural network with parameters θ, takes as input a target item $I_t \in \mathcal{I}$ and induces a message $M = w_1 ... w_l \in \mathcal{M}$, a sequence of tokens $w_i \in \mathcal{V}$:

$$M = A_{S_\theta}(I_t) = \text{RNN}_S(g(f_S(I_t))) \qquad (1)$$

where $f_S : \mathcal{I} \rightarrow \mathbb{R}^d$ is a function mapping an item to a d-dim vector, $g : \mathbb{R}^d \rightarrow \mathbb{R}^{d_{\text{RNN}}}$ is a linear layer, and RNN_S is a recurrent net (in our case a GRU (Cho et al., 2014)) with d_{RNN} input-dim. The RNN takes $g(f(I_t))$ as the initial context representation and generates message M token-by-token until an eos is produced or L tokens have been generated.

The receiver A_{R_ϕ}, a neural network with parameters ϕ, takes message M as input, along with a set of n items including the target $\{I_1, ... I_n\} \ni I_t$, and predicts which item is the target:

$$A_{R_\phi}(M, I_1, ..., I_n) = softmax\Big(\begin{bmatrix} f_R(I_1)^T \\ ... \\ f_R(I_n)^T \end{bmatrix} \times \text{RNN}_R(M) \Big) \qquad (2)$$

$f_R : \mathcal{I} \rightarrow \mathbb{R}^d$ is a function mapping an item to a d-dim vector. RNN_R is a recurrent net mapping a sequence of symbols to a d-dim vector. The output of the Receiver is thus a probability distribution over n input items, with the highest probability representing the item most likely to be the target.

To train the two agents, given a training set $\mathcal{D}_{\text{train}}$ which is a set of n-item tuples and the indices of

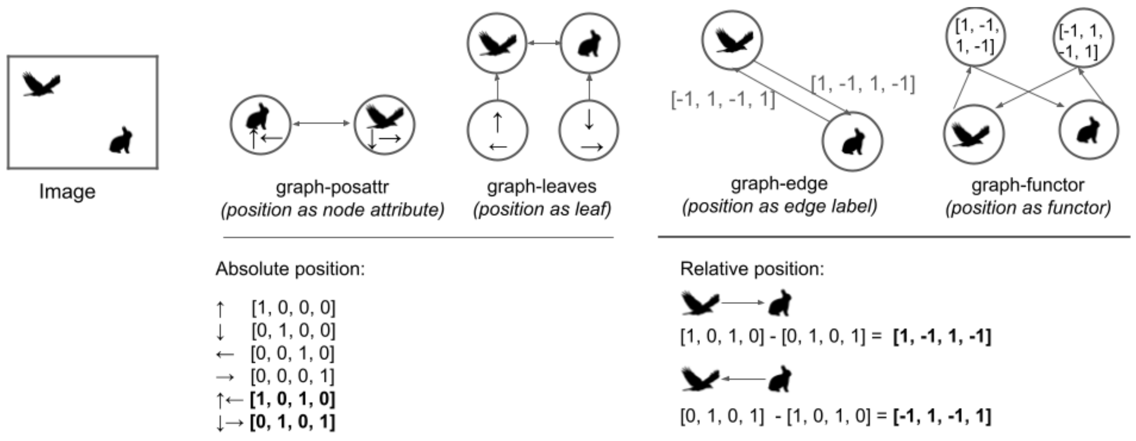

Figure 1: An item and four graph representations. The item depicts a natural scene which can be loosely described as "an eagle is flying over a rabbit to its left". In our simple item-description language, it is "eagle top left rabbit bottom right".

the target items, we minimize the following loss

$$L(\theta, \phi, \mathcal{D}_{\text{train}}) = - \sum_{(t, I_1, ..., I_n) \in \mathcal{D}_{\text{train}}} \log \left(A_{R_\phi}(A_{S_\theta}(I_t), I_1, ..., I_n) \right) \quad (3)$$

Note that, because of the discreteness of messages, this loss is not continuous with regards to the parameters of the sender, and thus minimizing the loss using gradient descent can only optimize the receiver. To optimize the sender, there are two widely used solutions. First, we can use the reinforcement learning method, in which a reward is given to the two agents when they together pass the game (Lazaridou et al., 2017). Second, we can "soften" the discreteness of messages using Gumbel-softmax to allow for back-propagation of errors through messages. In our experiments we opt for the latter, since it is simpler and achieves higher communication success (Havrylov and Titov, 2017).

Input Encoding

For the image representations, we use two-layer convolutional neural networks (unlike Lazaridou et al. (2018); Chaabouni et al. (2022), we did not find pre-trained networks e.g. ResNet beneficial).

For graph representations, we employ two types of graph neural networks, depending on whether edges are labeled or not. For graphs with edge labels (in *graph-edge* or "position as edge label"), we use two-layer GATv2Conv networks (Brody et al., 2022), which can capture edge labels in graph global representations. For the other graph representations, we use two-layer GCNConv networks (Kipf and Welling, 2017).

4 Experimental Setup

We implement our setup in Python, using the EGG library (Kharitonov et al., 2021) for the agents and Pytorch-geometric (Fey and Lenssen, 2019) for graph neural networks. Each experiment is performed with five different random seeds.

Data. We use a collection of 50 black shapes (see Appendix A), which we combine into 4-positional grids. From 14700 distinct items, we generate 11760 game rounds for training and 2940 game rounds for testing.

Agents. We experiment with the five input representations: image and four types of graphs described in Section 3.2. We report our hyper-parameter choices in Appendix B.

Game. We experiment with different game sizes (i.e. the number of items in each round, including the target and the distractors). Our game sizes are 2, 5, and 20. Our games also allow for different maximum message length; in particular: 2, 4, 6, and 10 tokens.

5 Communication Success

Communication success measures how well the two agents successfully solve the game together, i.e. how often the Receiver correctly selected the target described by the Sender. It is computed as the ratio of the number of successful game rounds

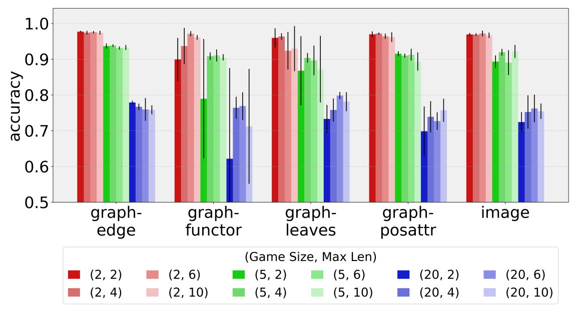

Figure 2: Test communication success rate for different game sizes and maximum message lengths.

over the total number of game rounds played (see, e.g. Lazaridou et al. (2017)).

This is the first metric that requires attention because if the communication success rate is low, the emergent language is not helping to solve the game, and is therefore uninteresting to us. Obviously, this metric is influenced by the difficulty of the game, which we can control with the number of distractors (such that a greater number of distractors leads to more difficult games).

Figure 2 shows communication success rate for different game sizes and maximum message lengths. As we can see, the rate drops when the game size increases. Nevertheless, even when there are 19 distractors, the (test) rate is often higher than 70%, much higher than the 5% rate that would be achieved with random guessing. This demonstrates the effectiveness of our agents as well as the used input representations. Overall the different input representations behave similarly in terms of communication success, although differences start to become noticeable for games with more distractors. The image and the graph-edge representations perform best overall, with the graph-edge representation performing better in larger game sizes. We also note that graph-functor and graph-leaves exhibit more variance between model runs, possibly due to the use of position-nodes (which is what distinguishes these two type of graphs from other graph encodings).

6 Compositionality

We are now ready to ask whether the emergent languages show any degree of compositionality. This is not a trivial question to answer, given that the messages are not readily interpretable to humans. The widespread approach in the field is to use a metric called Topographic Similarity (*topsim* from here on; Brighton and Kirby, 2006). The goal of this metric is to capture the similarity between the topologies of two sets of items. Mathematically, the topology of a set of items $\mathcal{I}$ is characterized by the distance metric d_I which measures dissimilarity between every two items. Similarly, we use the distance metric d_M to characterize message dissimilarity for the set of messages $\mathcal{M}$. The *topsim* metric is calculated as the negative Spearman correlation ρ between the two lists $(d_I(I_i, I_j))$ and $(d_M(A_S(I_i), A_S(I_j)))$ where $I_i, I_j \in \mathcal{I}$.

An attractive property of *topsim* is that it is agnostic to the type of input, as long as it can be characterized by a distance metric. A common choice is *cosine distance* for d_I and *minimum edit distance* for d_M (see e.g. Chaabouni et al. (2022)). Here, we note that we can actually make use of this degree of freedom to further explore the type of compositionality (or lack of) of the emergent languages.

As explained above, *topsim* measures the topological similarity between two sets; therefore, if one set is compositional due to factor A, a high *topsim* value should imply that the other set should also be compositional due to the same factor A. In other words, *topsim* will only capture compositionality in the message space if $d_I(I)$ provides a topological ordering of items that is based on the same feature(s) that were decomposed in the messages. This has implications for the choice of I and d_I; in particular, we observe that the usual method applied over image input (cosine distance over image embeddings) is not guaranteed to capture compositionality over conceptual properties of the input, since the items that are deemed more similar according to cosine distance may be so due to coincidences at the raw pixel level (which could even be noise, Bouchacourt and Baroni, 2018). Thus, we opt for a choice of I that allows us to interpret what are the compositional features that we are measuring –as done in other work that uses semantic attributes for I (Lazaridou et al., 2018; Chaabouni et al., 2020).

To this aim, we design a script-like representation that can be easily manipulated to incorporate or remove certain features, thus influencing the topology created with d_I (for which we use *Hamming distance*). We refer to this representation as *item-script*, and provide examples in the upcoming sections and in Appendix C.

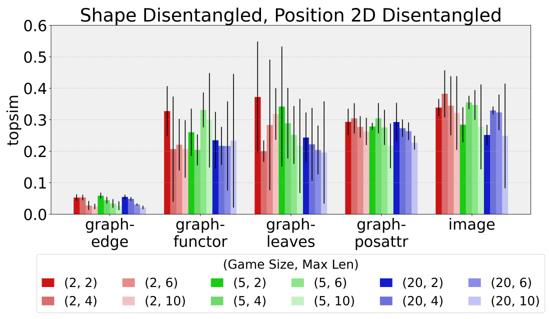

Figure 3: *Topsim* between the emergent languages and the fully-compositional item-script (on test data).

6.1 Do agents develop fully compositional languages?

We first check whether agents develop the most complex version of a language to describe our items. This would be a language with dedicated tokens for the shape and position of each entity. For this case, the item-script looks very similar to a simplified version of English; for instance, the items in Figure 1 would be described as "eagle top left rabbit bottom right". Thus, the topology of the input will be organized according to these units, such that the distance between "eagle top left rabbit bottom right" and "butterfly top left rabbit bottom right" is 1, while it becomes 2 between the former and "butterfly bottom left rabbit bottom right".

As we can see in Figure 3, the languages emerging from image-based communication and graph-based communication exhibit a similar, moderate degree of compositionality, with the exception of graph-edge. This result is somewhat unexpected, since it seems at odds with the findings in Lazaridou et al. (2018). In that paper, the authors observed that the language emerging from the image setup is less compositional than the language emerging from feature-based input (which is disentangled, as is the case for graphs). We now perform further analyses to investigate the source of this result.

6.2 Where does the composition come from?

To address this question, we manipulate the topology of the input to find out which features are used compositionally.

Shape vs. Position Our first manipulation is an ablation, of either shape or position. To achieve that, we take the fully compositional item-description language described above, and we remove either shape or position information. Continuing with the example above, the item would be described as "eagle rabbit" when omitting the position, and "top left bottom right" when omitting the shape information. Therefore, the topology $d_I(I)$ will reflect only these features. We refer to these two conditions as Shape Disentangled and Position2D Disentangled, respectively.

Figure 4-column 1 shows the *topsim* for the inputs described above. It is clear that the "graph-edge" agents achieve superior *topsim* when the item-description language is fully disentangled over shape-information only (top graph), but near-zero *topsim* when shapes are omitted (bottom graph). This suggests that compositionality on the emerged languages is mostly driven by shape rather than position. For image-based agents and the other graph-based agents we come to the opposite conclusion. That is, in configurations other than graph-edge, *topsim* is remarkably low in the case of Shape Disentangled (Figure 4-column 1-top) and higher for Position 2D Disentangled (Figure 4-column 1-bottom), suggesting that compositionality in these configurations is focused on position rather than shape. Since "graph-edge" is the only encoding that does not use nodes to encode positions, a tentative interpretation is that the model focuses on information represented in nodes more than it does so for information in edges.

We then considered whether agents may use 1-dimensional positions instead; that is, a single token to denote each of the 4 positions in the grid, referring to: "top-left", "top-right", "bottom-left", "bottom-right". Thus we measured *topsim* over an input-script using 1-dimensional positions (and no shape information); we refer to this script as Position1D Disentangled. As can be seen in Figure 4-column 2, this analysis reveals that the emergent languages are consistent with this form of compositionality as well, to a similar extent to 2D (only slightly higher). The most straightforward interpretation of this result is that agents combine both types of positional information (1D and 2D), but we must bear in mind that some overlap is expected between these two topologies.

Entity Decomposition The input topology used in the previous analysis assumes that shape or position are decomposed for each of the two entities in the target. However, this need not be the case: agents could use a single token to refer to every combination of two shapes (e.g. one token for ev-

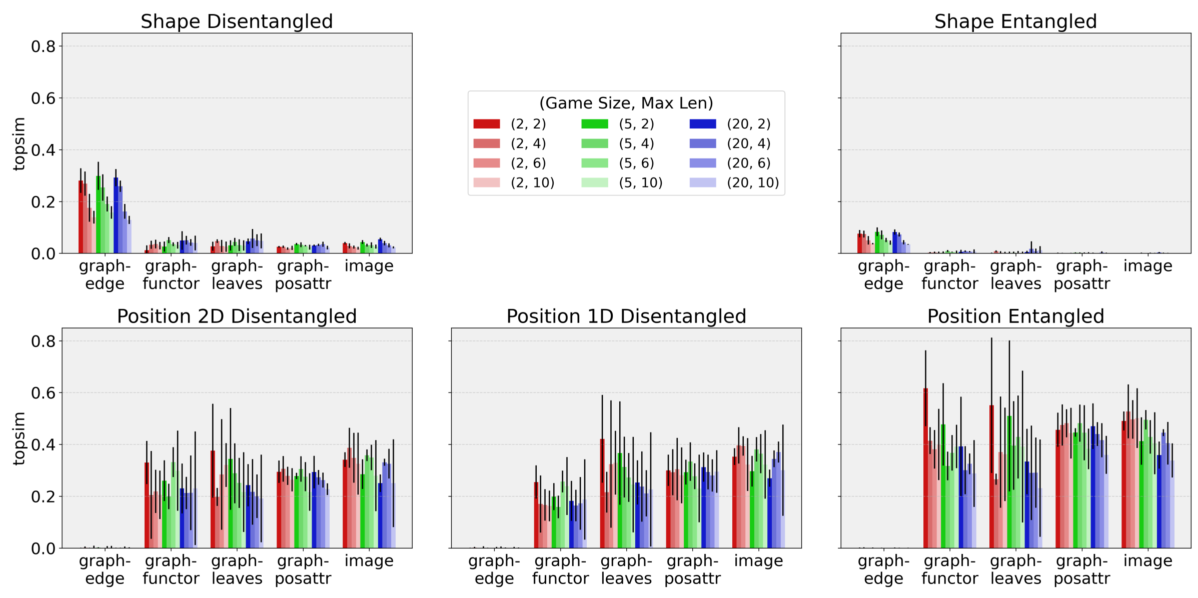

Figure 4: *Topsim* between the emergent languages and the shape-only or position-only item-description languages (test results). The first row is shape-only, and the second row is position-only.

ery target with an eagle and a rabbit), or for every grid configuration (e.g. use the same token whenever the two shapes are on the bottom row). We now analyze whether agents used a mixed strategy in which either shape or position are used compositionally, but not both.

The *topsim* values for this experiment are shown in Figure 4-column 3. Comparing entanglement and disentanglement gives us interesting insights. To begin with, all the emergent languages favor disentangled shapes as opposed to entangled, as evidenced with higher *topsim* in row1-column1 vs. row1-column3 (note that values are rather low overall in these graphs due to the omission of positional information). However, when it comes to position, emergent languages favor entangled positions (higher *topsim* in row 2-column 3, compared to other columns in the same row), suggesting that messages tend to incorporate a description of the full grid configuration rather than describing the position of each shape.

7 Language Evolution

The languages emerging from our agents evolve over time. Figure 5 shows the evolving compositionality (i.e. *topsim* for shape-disentangled and position2d-disentangled item-script) and communication success of five models trained on the same game.

As training progresses, we observe an increase in communication success. Interestingly, even though we observed a preference to decompose positions rather than shapes for image-based and graph-functor/leaves/posattr-based agents, this analysis reveals that this preference is not constant over time; in fact, the composition over position decreases over time while compositionality over shapes increases, albeit slightly.

8 Discussion

Altogether, our analyses suggest that our games are more challenging than traditional single-entity games, at least in terms of analyzing compositionality. As we showed in section 6, multiple factors can potentially be decomposed; in particular, shape and position. In our game, shape is the identity factor of entities (that is, that is the only property that is distinct between entities), while position can be seen as an attribute of an entity (such that the same entity can be in different positions accross targets). In a game with a single entity, compositionality would be limited to decomposing shape and position. However, in a multi-entity game, this does not suffice, since even a perfectly compositional language requires an additional mechanism to link each entity (shape) with their positional attribute. In natural languages, this may be achieved with word order or case marking. The moderate scores *topsim* scores that we observe for Disentangled conditions (paired with the high accuracy scores) suggest that some initial instances of this basic level of syntax may be emerging, but the higher

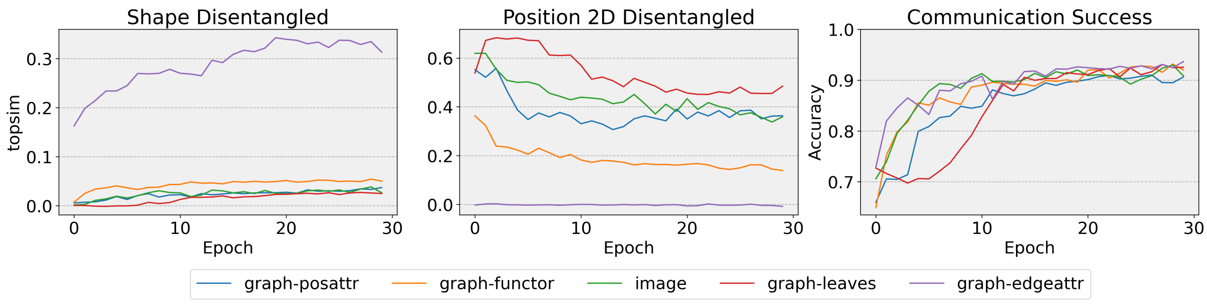

Figure 5: *Topsim* and communication success rate during learning (measured over test data), for representative runs on a game of size 5 and maximum message length 4.

scores for Position Entangled indicate that agents occasionally conflate the positions of both entities in one single token.

In our view, this showcases that multi-entity games provide an interesting and linguistically relevant challenge to tackle. We expected that graph representations, which can transparently represent which features (positions) connect to which entities (shapes), would be better suited for the emergence of this basic level of syntax; however, in our simulations we found that images are similarly competitive (and even better than some of the graph schemes). This is not entirely surprising in the case of compositionality over entangled positions (for which we do not anticipate any advantage from graphs) but it is somewhat unexpected for disentangled positions. We must note that the simplicity of our grid-based targets may have obscured the advantage of providing structure representations in graphs, since the image model could likely learn a similarly structured representation from our black-and-white images with clearly separated shapes. Thus, it is entirely possible that our current setup did not manage to fully exploit advantages of graphs. We expect these become more relevant when extending our game to more visually complex targets holding a varying number of entities per target.

Limitations

We have explored image and graph representations separately, such that each model only had access to one or the other type of representation. However, in our view these representations have different cognitive interpretations: images can be thought of as low-level sensory input, while graphs —which are a higher-level abstract construct, involving more structure— could be interpreted as part of our conceptual system. Thus, these representations are not mutually exclusive, and may play different (perhaps complementary) roles in language emergence.

We have limited our game to two-agent interaction, although recent work has shown the necessity of having multiple generations (Ren et al., 2020) and a larger population (Rita et al., 2022). However, as we emphasized in Section 3.3, our setup can be seamlessly integrated into existing agent-based architectures, which facilitates future extension into multiple-generation-based and population-based frameworks.

Our analyses of compositionality is limited to the use of *topsim*. We have opted for this due to the flexibility of this metric when it comes to its input (which allowed us to perform an ablation analysis); however, there are other metrics in the literature that we have not applied and could potentially bring further insights (e.g. Andreas, 2019; Chaabouni et al., 2020).

Finally, all our experiments are done using the same number of entities across items. This is a useful simplification for a first exploration of multi-entity games, so that our compositionality analysis remained tractable and insightful. However, we may have observed more variation in the performance of the representations we explored –in particular, we expect that the usefulness of graphs becomes more evident when using a greater and more variable number of entities, but the empirical investigation is left to future work.

Ethical Considerations

Development and testing of the models was performed on a desktop computer using CPU, with trainable model parameter counts ranging from 140000 for graph-based games and 220000 for image-based games. Each model train and test loop takes approximately 30 minutes to complete on average, except for images which can take twice as long. The total system power used is in the order of 150W, and the reported simulations cover 300 models for an estimated total power consumption of 27 kWh. In conclusion, the ecological impact of this project is relatively small compared to examples such as large language models. We do not expect any other potential risks as a result of our research.

References

Jacob Andreas. 2019. Measuring Compositionality in Representation Learning. In *International Conference on Learning Representations*.

Diane Bouchacourt and Marco Baroni. 2018. How agents see things: On visual representations in an emergent language game. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 981–985, Brussels, Belgium. Association for Computational Linguistics.

Henry Brighton and Simon Kirby. 2006. Understanding Linguistic Evolution by Visualizing the Emergence of Topographic Mappings. *Artificial Life*, 12(2):229–242.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. 2020. Compositionality and Generalization In Emergent Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442, Online. Association for Computational Linguistics.

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

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

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. 2018. Emergent Communication in a Multi-Modal, Multi-Step Referential Game. In *International Conference on Learning Representations*.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Serhii Havrylov and Ivan Titov. 2017. Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. 2017. Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark. Association for Computational Linguistics.

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

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. 2018. Emergence of Linguistic Communication from Referential Games with Symbolic and Pixel Input. In *6th International Conference on Learning Representations*.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-Agent Cooperation and the Emergence of (Natural) Language. In *International Conference on Learning Representations*.

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

Yuchen Lian, Arianna Bisazza, and Tessa Verhoef. 2023. Communication Drives the Emergence of Language Universals in Neural Agents: Evidence from the Word-order/Case-marking Trade-off. *Transactions of the Association for Computational Linguistics*, 11:1033–1047.

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

Barbara H Partee. 2008. *Compositionality in formal semantics: Selected papers*. John Wiley & Sons.

Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B Cohen, and Simon Kirby. 2020. Compositional languages emerge in a neural iterated learning model. In *International Conference on Learning Representations*.

Mathieu Rita, Florian Strub, Jean-Bastien Grill, Olivier Pietquin, and Emmanuel Dupoux. 2022. On the role of population heterogeneity in emergent communication. In *International Conference on Learning Representations*.

Brian Skyrms. 2010. *Signals: Evolution, Learning, and Information*. Oxford University Press, Oxford, GB.

Agnieszka Slowik, Abhinav Gupta, William L. Hamilton, Mateja Jamnik, Sean B. Holden, and Christopher J. Pal. 2020. Exploring structural inductive biases in emergent communication. *CoRR*, abs/2002.01335.

Luc L. Steels. 2015. *The Talking Heads experiment*. Number 1 in Computational Models of Language Evolution. Language Science Press, Berlin.

Agnieszka Słowik, Abhinav Gupta, William L. Hamilton, Mateja Jamnik, and Sean B. Holden. 2020. Towards graph representation learning in emergent communication. *Preprint*, arXiv:2001.09063.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24.

A Data generation

We generate 2x2 grids, i.e. grids with four positions. In each grid, two positions are filled with shapes. A shape is never repeated in the same grid.

We use a total of 50 different shapes. We generate every possible combination of 2 shapes (excluding repetitions). This adds up to $\frac{50^2-50}{2} = 1225$ combinations. We then generate all the possible grids for each pair (i.e. for eagle and rabbit, we would generate N grids, permuting all the positions that these two shapes can appear in within a grid). At this stage, we represent every target (i.e. grid and shape combination) as a string with the format "eagle_0_0_rabbit", such that 0 is a placeholder for an empty position, and relative order of elements in the string indicates position (i.e. top left: eagle, top right: empty, bottom left: empty, bottom right: rabbit).

This results in a dataset of 14700 items. The strings can then be converted into images or graphs. For images, shapes corresponding to the relevant shape and position are placed on a white background. For graphs, shapes are encoded as nodes, while positions may be encoded as nodes, node attributes, or edge attributes, as shown in Figure 1.

B Hyper-parameters

Relevant hyper-parameters for all experiments are shown in Table 1. Game size indicates the amount of distractors plus the target. Gumbel-softmax temperature controls the Gumbel-softmax sampling distribution: lower values tend towards a one-hot encoding, whereas higher values tend towards a uniform encoding. The initial learning rate is adjusted with Adam. We expect the rest of hyper-parameters to be self-explanatory.

Parameter	**Value(s)**
batch size	32
initial learning rate	1e-3
number of epochs	30
train data ratio	0.8
game size	[2, 5, 20]
max length	[2, 4, 6, 10]
vocabulary size	100
RNN cell	GRU
RNN hidden size	80
image size	120×120
Gumbel-softmax temperature	1.0
trainable temperature	True

Table 1: Hyper-parameters explored in our simulations.

C Item-Script

As explained in section 6, we use an item-script representation for the input I to *topsim*. Table 2 shows an example of each item-script, for the item depicted in Figure 1.

Input-script	**Example**
shape-disentangled position2d-disentangled	eagle top left rabbit bottom right
shape-disentangled	eagle rabbit
shape-entangled	eagle-rabbit
position2d-disentangled	top left bottom right
position1d-disentangled	top-left bottom-right
position-entangled	top-left-bottom-right

Table 2: Examples of the item-script that we use to represent the input to *topsim*.