Speaking Your Language: Spatial Relationships in Interpretable Emergent Communication

Olaf Lipinski[1*] **Adam J. Sobey**[2,1] **Federico Cerutti**[3] **Timothy J. Norman**[1]
[1]University of Southampton [2]The Alan Turing Institute [3]University of Brescia
{o.lipinski,t.j.norman}@soton.ac.uk
asobey@turing.ac.uk
federico.cerutti@unibs.it

Abstract

Effective communication requires the ability to refer to specific parts of an observation in relation to others. While emergent communication literature shows success in developing various language properties, no research has shown the emergence of such positional references. This paper demonstrates how agents can communicate about spatial relationships within their observations. The results indicate that agents can develop a language capable of expressing the relationships between parts of their observation, achieving over 90% accuracy when trained in a referential game which requires such communication. Using a collocation measure, we demonstrate how the agents create such references. This analysis suggests that agents use a mixture of non-compositional and compositional messages to convey spatial relationships. We also show that the emergent language is interpretable by humans. The translation accuracy is tested by communicating with the receiver agent, where the receiver achieves over 78% accuracy using parts of this lexicon, confirming that the interpretation of the emergent language was successful.

1 Spatial referencing in emergent communication

Emergent communication allows agents to develop bespoke languages for their environment. While there are many successful examples of efficient (Rita et al., 2020) and compositional (Chaabouni et al., 2020) languages, they often lack fundamental aspects seen in human language, such as syntax (Lazaridou and Baroni, 2020) or recursion (Baroni, 2020). It is argued that these aspects of communication are important to improve the efficiency and generalisability of emergent languages (Baroni, 2020; Boldt and Mortensen, 2024; Rita et al., 2024). However, the current architectures, environments, and reward schemes are yet to exhibit such fundamental properties.

One such aspect is the development of *deixis* (Rita et al., 2024), which has been described as a way of pointing through language. Examples of *temporal deixis* include words such as "yesterday" or "before," and *spatial deixis* include words such as "here" or "next to" (Lyons, 1977). In emergent communication, Lipinski et al. (2023) investigate how agents may refer to repeating observations, which could also be viewed from the linguistic perspective as investigating *temporal deixis*. However, while there are advocates to investigate how emergent languages can develop key concepts from human language (Rita et al., 2024), no work has demonstrated the emergence of relative references to specific locations *within* an observation, or *spatial deixis*.

Spatial references would be valuable in establishing shared context between agents, increasing communication efficiency by reducing the need for detailed descriptions, and adaptability, by removing the need for unique references per object. For example, instead of describing a new, previously

*Corresponding author: o.lipinski@soton.ac.uk

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-4446

unseen object, such as "a blue vase with intricate motifs on the table," one could simply use spatial relationships and say "the object left of the plate." Spatial referencing streamlines communication by leveraging the shared environment as a reference point. In dynamic environments where objects might change positions, spatial references enable agents to easily track and refer to objects without having to update their descriptions. This enhances communication efficiency and improves interaction and collaboration between agents. These elements may also help the evolved language become human interpretable, allowing the development of trustworthy emergent communication (Lazaridou and Baroni, 2020; Mu and Goodman, 2021).

This paper therefore explores how agents can develop communication with spatial references. While Rita et al. (2024) posit that the emergence of these references might require complex settings, we show that even agents trained in a modified version of the simple referential game (Lazaridou et al., 2018; Lewis, 1969) can develop spatial references.[2] This resulting language is segmented and analysed using a collocation measure, Normalised Pointwise Mutual Information (NPMI) adapted from computational linguistics. NPMI allows us to measure the strength of associations between message parts and their context, making it a valuable tool for gaining insights into the underlying structure of the emergent language. Using NPMI, we show how the agents compose such spatial references, providing the first hint of a syntactic structure, and showing that the emergent language can be interpreted by humans.

2 Development of a spatial referential game

Current emergent communication environments have not produced languages incorporating spatial references. To address this, we present a referential game (Lazaridou et al., 2018) environment where an effective language requires communication about spatial relationships.

2.1 Referential game environment

In the referential game, there are two agents, a sender and a receiver. The sender observes a vector and transmits its compressed representation through a discrete channel to the receiver. The receiver observes a set of vectors and the sender's message. One of these vectors is the same as the one the sender has observed. The receiver's goal is to correctly identify the vector the sender has described, among other vectors referred to as distractors. The simplicity of the referential games enables the reduction of extraneous factors which could impact the emergence of spatial references, such as transfer learning of the vision network or exploring action spaces in more complex environments.

In this work, the sender's input is an observation in the form of a vector $\boldsymbol{o} = [o_1, o_2, o_3, o_4, o_5]$, where $\forall o \in \{-1, 0, 1 \dots 59\}$. The vector $\boldsymbol{o}$ is always composed of 5 integers. The observation includes a -1 in only one position, *e.g.*, $\boldsymbol{o}_3 = -1$ for $\boldsymbol{o} = [x, x, -1, x, x]$, to indicate the target integer for the receiver to identify. $\boldsymbol{o}$ represents a window into a longer sequence $\boldsymbol{s}$, which is randomly generated using the integers $\{0 \dots 59\}$ without repetitions. This sequence is visible to the receiver, but **not** to the sender. As the target's position in the sequence is unknown to the sender, it has to rely on the relative positional information present in its observation, necessitating the use of *spatial referencing*.

Due to the window into the sequence being of length 5, it is necessary to shift the window when it approaches either extent of the sequence. The window is then shifted to the other side, maintaining the size of 5. For example, given a short sequence $\boldsymbol{s} = [7, 5, 2, 12, 10, 4, 3, 15, 16, 13, 14, 6, 9, 8, 11, 1]$, if the selected target is 1, since there are no integers to the right of 1 the vector $\boldsymbol{o}$ would be $\boldsymbol{o} = [6, 9, 8, 11, -1]$ where it is shifted to the left as it approaches this rightmost extent of the sequence.

Due to the necessity of maintaining the window size, some observations provide additional positional information to the sender agent. Given the same example sequence $\boldsymbol{s}$, we can categorise all observations into 5 types. The *begin* and *begin+1*, where the target integer is either at, or one after, the beginning of the sequence, *i.e.*, $\boldsymbol{o} = [-1, 5, 2, 12, 10]$ or $\boldsymbol{o} = [7, -1, 2, 12, 10]$. The *end* and *end-1*, where the target integer is either at, or one before, the end of the sequence, *i.e.*, $\boldsymbol{o} = [6, 9, 8, 11, -1]$ or $\boldsymbol{o} = [6, 9, 8, -1, 1]$. The most common case is the *middle* observation, where the target integer is anywhere in the sequence, excluding the first, second, second to last, and last positions, *e.g.*, $\boldsymbol{o} = [12, 10, -1, 3, 15]$. Given a window of length 5, only 4 specific target integer positions per sequence can result in the other observations (*begin*, *begin+1*, *end-1*, and *end*). All other target

[2] Our code is available on GitHub at https://github.com/olipinski/TPG

integer positions within the sequence fall into the *middle* category, as they do not occupy the first, second, second to last, or last positions. Consequently, the majority of the target integer positions result in a *middle* type observation.

The sender's output is a message defined as a vector $\boldsymbol{m} = [m_1, m_2, m_3]$, where $m \in \{1 \dots 26\}$. 26 is chosen to allow for a high degree of expressivity, with the agents being able to use over 17k different messages, while also matching the size of the Latin alphabet. Since such a vocabulary size is enough to convey any information in natural languages like English, we consider that this should also apply to the agents. The vector $\boldsymbol{m}$ is always composed of 3 integers.

The receiver's input is an observation consisting of three vectors: the sender's message $\boldsymbol{m}$, the sequence $\boldsymbol{s}$, and the set of distractor integers together with the target integer $\boldsymbol{td}$. The distractor integers are randomly generated, without repetitions, given the same range of integers as the original sequence $\boldsymbol{s}$, *i.e.*, $\{0 \dots 59\}$, excluding the target object itself. Given an environment with 3 distractors, $\boldsymbol{td}$ could be $[d_1, t, d_2, d_3]$, where t is the target object and d_1, d_2, d_3 are distractor objects. The position of the target object in $\boldsymbol{td}$ is randomised.

For example, given the sequence $\boldsymbol{s} = [7, 5, 2, 12, 10, 4, 3, 15, 16, 13, 14, 6, 9, 8, 11, 1]$, and the sender's observation $\boldsymbol{o} = [4, 3, -1, 16, 13]$, the vector $\boldsymbol{td}$ could be $\boldsymbol{td} = [7, 15, 11, 9]$, with 15 being the target that the receiver needs to identify. The sender could produce a message $\boldsymbol{m} = [3, 1, 1]$, which would mean that the target integer is one after the integer 3. This message would then be passed to the receiver, together with $\boldsymbol{s}$ and $\boldsymbol{td}$. The receiver would then have to correctly understand the message $\boldsymbol{m}$ (*i.e.*, that the target is one after 3) and find the integer 3 together with the following integer in the sequence $\boldsymbol{s}$. Having identified the target 15 given the message $\boldsymbol{m}$ and the sequence $\boldsymbol{s}$, it would output the correct position of this target in the $\boldsymbol{td}$ vector, *i.e.*, 2, since $\boldsymbol{td}_2 = 15$.

2.2 Spatial reference formalisation

To provide a generalisation of our results, we formalise what we refer to as spatio-temporal references. Let O represent an abstract observation that an agent perceives from its environment, $O \in \mathbb{R}^m$, where m represents the dimensions of the observation. For a 3D observation, m could be $m = j \times k \times d$. Such an m could represent a $j \times k$ matrix of $d = 3$ values, which, for example, could be an RGB picture, with $j \times k$ pixels and one value for each of the RGB colours ($d = 3$). The m dimensions can represent the spatial, temporal, or other positions.

Let O_p and O_t be the coordinates of some elements in O, represented by an m-tuple of natural numbers $(x_1, x_2 ... x_m)$ and $(y_1, y_2 ... y_m)$, respectively. O_p represents the reference point and O_t represents a target point.

Then, the relative distance function $d(O_p, O_t)$ returns an m-tuple of integers $(z_1, z_2 ... z_m)$, such that $z_i = x_i - y_i$. This relative distance function allows for unambiguous identification of the target object O_t, given that the position of O_p is known.

We define the spatio-temporally referent expression as a mapping of the value of $d(O_p, O_t)$, the reference point O_p, and their context O, to a specific linguistic or symbolic phrase that describes the relationship between O_p and O_t. This mapping can be represented as:

$(O, d(O_p, O_t), O_p) \rightarrow \text{Phrase}(O, d(O_p, O_t), O_p)$

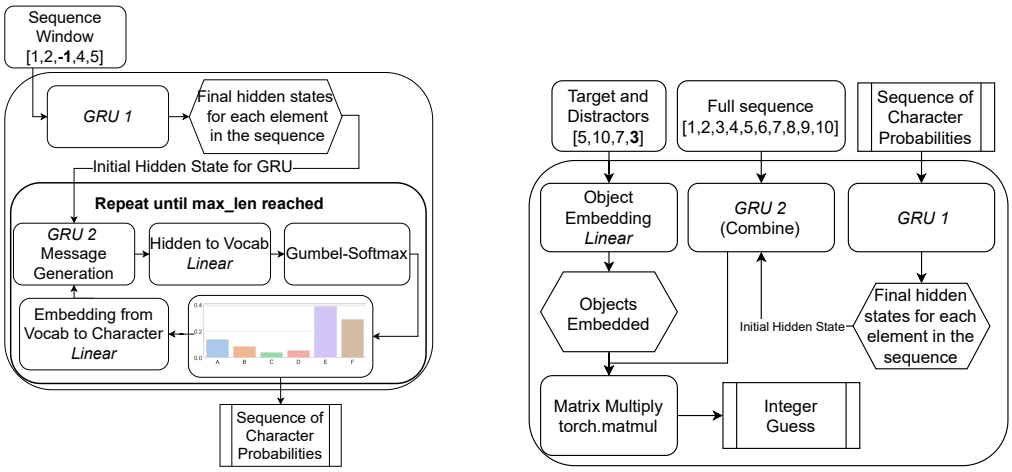
where the resulting expression $\text{Phrase}(O, d(O_p, O_t), O_p)$ is a description of the reference point O_p and its relative distances to the target point O_t, given the context O.

The version of spatial referencing in our environment is a specific case of the general spatial reference formalisation, where the observation O is represented as a one-dimensional tensor, and the target point O_t is always indicated by the value -1 within the tensor. The sender's task is to describe the relative position of the target O_t within this sequence, using a message that effectively communicates the spatial relationship between a chosen O_p and the target O_t.

3 Agent Architecture

The agent architecture follows that of the most commonly used EGG agents (Kharitonov et al., 2019). This architecture is used to maintain consistency with the common approaches in emergent communication research (Chaabouni et al., 2019, 2020; Kharitonov et al., 2019; Lipinski et al., 2023;

 https://doi.org/10.52202/079017-4446



(a) The sender architecture.

(b) The receiver architecture.

Figure 1: The sender and receiver architectures. Adapted from (Lipinski et al., 2023).

Ueda and Washio, 2021), increasing the generalization of the results presented in this work. All environmental observations, *i.e.*, $\boldsymbol{o}$, $\boldsymbol{s}$, and $\boldsymbol{td}$, are passed in as scalars, as one-hot encoding of the observation vectors leads to agents memorising the dataset.

The sender agent, shown in Figure 1a, receives a single input, the vector $\boldsymbol{o}$, which is passed through the first GRU of the sender. The resulting hidden state is used as the initial hidden state for the message generation GRU (Cho et al., 2014). The message generation GRU is used to produce the message, character by character, using the Gumbel-Softmax reparametrization trick (Jang et al., 2017; Kharitonov et al., 2019; Mordatch and Abbeel, 2018). The sequence of character probabilities generated from the sender is used to output the message $\boldsymbol{m}$.

$\boldsymbol{m}$ is input to the receiver agent, shown in Figure 1b, together with the full sequence $\boldsymbol{s}$ and the target and distractors $\boldsymbol{td}$. The message is processed by the first receiver GRU, which produces a hidden state used as the initial hidden state for the GRU processing the sequence $\boldsymbol{s}$. This is the only change from the standard EGG architecture (Kharitonov et al., 2019). This additional GRU allows the receiver agent to process the additional input sequence $\boldsymbol{s}$, using the information contained within the message $\boldsymbol{m}$. The goal of this GRU is to use the information provided by the sender to correctly identify which integer from the sequence $\boldsymbol{s}$ is the target integer. The final hidden state from the additional GRU is multiplied with an embedding of the targets and distractors, to output the receiver's prediction. This prediction is in the form of the index of the target within $\boldsymbol{td}$.

Following the commonly used approach (Kharitonov et al., 2019), agent optimisation is performed using the Gumbel-Softmax reparametrization (Jang et al., 2017; Mordatch and Abbeel, 2018), allowing for direct gradient flow through the discrete channel. The agents' loss is computed by applying the cross entropy loss, using the receiver target prediction and the true target label. The resulting gradients are passed to the Adam optimiser and backpropagated through the network. Detailed training hyperparameters are provided in Appendix A.

4 Message interpretability and analysis using NPMI

To analyse spatial references in emergent language, a way to identify their presence is essential. In discrete emergent languages, interpretation is typically done by either using dataset labels in natural language (Dessì et al., 2021), or by qualitative analysis of specific messages (Havrylov and Titov, 2017). However, both of these techniques require message-meaning pairs, and so neither would be able to identify the presence of spatial references, as the labels for spatial relationships that the agents refer to would not necessarily be available. One approach that could overcome this problem is emergent language segmentation using Harris' Articulation Scheme, recently employed by Ueda et al. (2023). Ueda et al. (2023) compute the conditional entropy of each character in the emergent language, segmenting the messages where the conditional entropy increases. However,

even after language segmentation, there is no easy way to interpret the segments, as no method has been proposed to map them to specific meanings.

We present an approach to both segment the emergent language and map the segments to their meanings. We use a collocation measure called Normalised Pointwise Mutual Information (NPMI) (Bouma, 2009), often used in computational linguistics (Lim and Lauw, 2024; Thielmann et al., 2024; Yamaki et al., 2023). It is used to determine which messages are used for which observations and to analyse how the messages are composed, including whether they are trivially compositional (Korbak et al., 2020; Perkins, 2021; Steinert-Threlkeld, 2020). By applying a collocation measure to different parts of each message as well as the whole message, we can address the problems of both segmentation and interpretation of the message segments. This approach allows any part of the message to carry a different meaning. For example, if an emergent message contains segments that frequently appear in contexts involving specific integers, NPMI can help identify these segments and their meanings based on their statistical association with those integers.

NPMI is a normalised version of the Pointwise Mutual Information (PMI) (Church and Hanks, 1989), which is a measure of association between two events. PMI is widely used in computational linguistics, to measure the association between words (Han et al., 2013; Paperno and Baroni, 2016). Normalising the PMI measure results in its codomain being defined between -1 and 1, with -1 indicating a purely negative association (*i.e.*, events **never** occurring together), 0 indicating no association (*i.e.*, events being **independent**), and 1 indicating a purely positive association (*i.e.*, events **always** occurring together). Normalised PMI is used for convenience when defining a threshold at which we consider a message or n-gram to carry a specific meaning, as the threshold can be between 0 and 1, instead of unbounded numbers in the case of PMI. [3]

To determine which parts of each message are used for a given meaning, two algorithms are proposed.

1. PMI_{nc} The algorithm to measure non-compositional monolithic messages, most often used for target positional information (*e.g.*, *begin+1* (Section 2)); and
2. PMI_{c} the algorithm to measure trivially compositional messages and their n-grams, used to refer to different integers in different positions.

A visual representation of the different types of messages that the algorithms can identify is provided in Figure 2. The PMI_{nc} algorithm can identify any non-compositional messages, while the PMI_{c} algorithm identifies both position variant and invariant compositional messages. The positional variance of the emergent language means that the position of an n-gram in the message also carries a part of its meaning. In this work, n-grams refer to a contiguous sequence of n integers from the sender's message. Consequently, in one message there are 3 unigrams (m_1, m_2, m_3), two bigrams ([m_1, m_2], [m_2, m_3]), and one trigram (*i.e.*, the whole message [m_1, m_2, m_3]).

Figure 2 shows that in the position invariant case, the bigram $[5, 6]$ always carries the meaning of 4. While in the position variant case, the bigram $[5, 6]$ in position 1 of the message means 4, but $[5, 6]$ in position 2 of the message means 8. This can also be interpreted as the position of the bigram containing additional information, meaning a single "word" could be represented as a tuple of the bigram and its position in the message, as both contribute to its underlying information. Non-compositional messages are monolithic, *i.e.*, the whole message carries the entire meaning. For example, message $[5, 6, 8]$ means the target is in the first position, while $[5, 6, 6]$ means the target is one to the right of 9, even though the two messages share the bigram $[5, 6]$.

The PMI_{nc} algorithm The PMI_{nc} algorithm calculates the NPMI per message by first building a dictionary of all counts of each message being sent, together with an observation that may provide positional information (*e.g.*, *begin+1*) or refer to an integer in a given position (*e.g.*, 1 left of the target). The counts of that message and the counts of the observation, including the integer position, are also collected. For example, consider the observation $\boldsymbol{o} = [4, -1, 15, 16, 13]$. For the corresponding message $\boldsymbol{m}$, the counts for each integer in each position relative to the target would increase by 1 (*i.e.*, $left1[4] += 1$, $right1[15] += 1$ *etc.*). The count for the message signifying *begin+1* would also be increased. Given these counts, the algorithm then estimates the probabilities of all respective events (messages, positional observations, and integers in given positions) and calculates the NPMI measure.

[3] Our implementation of NPMI is not numerically stable due to probability approximation, sometimes exceeding the [-1,1] co-domain. We provide more details in the code.

https://doi.org/10.52202/079017-4446

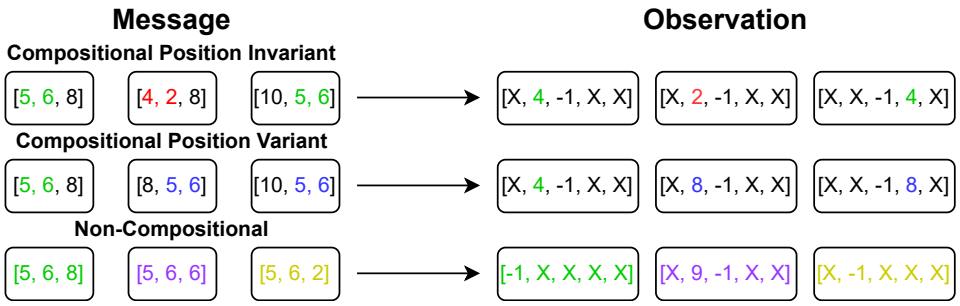


Figure 2: Examples of the different types of message compositionality that are possible to identify using the PMI algorithms.

The PMI$_c$ algorithm The PMI$_c$ algorithm first creates a dictionary of all possible n-grams, given the message space (m) and maximum message length (3). The list of all possible n-grams is pruned to contain only the n-grams present in the agents' language, avoiding unnecessary computation in the later parts of the algorithm. Given the pruned list of n-grams, the algorithm checks the context in which the n-grams have been used. The occurrence of each n-gram is counted, together with the n-gram position in the messages and the context in which it has been sent, or the integers in the observation. The n-gram position in the message is considered to account for the possible position variance of the compositional messages.

Consider the previous example, with $\boldsymbol{o} = [4, -1, 15, 16, 13]$ and a message $\boldsymbol{m} = [11, 13, 5]$. For all n-grams ($[11], [13], [5], [11, 13]$, *etc.*) of the message, all integers are counted, irrespective of their positions (*i.e.*, $counts[4]+ = 1$, $counts[15]+ = 1$, *etc.*).

Given these counts, the PMI$_c$ algorithm estimates the NPMI measure for all n-grams and all integers in the observations. These probabilities are estimated from the dataset using the count of their respective occurrences divided by the number of all observations/messages.

Once the NPMI measure is obtained for the n-gram-integer pairs, the algorithm calculates the NPMI measure for n-grams and referent positions or the positions of the integer in the observation the message refers to. For example, given an observation $\boldsymbol{o} = [4, -1, 15, 16, 13]$, if the message contains an n-gram which has been identified as referring to the integer 15, the rest of the message (*i.e.*, the unigram or bigram, depending on the length of the integer n-gram) is counted as a possible reference to that position, in this case, to position $right1$, or 1 to the right of the target. This procedure follows for all messages, building a count for each time an n-gram was used together with a possible n-gram for an integer. These counts are used to calculate the NPMI measure for n-gram and position pairs.

The PMI$_c$ algorithm also accounts for the possible position invariance of the n-grams, *i.e.*, where in the message the n-gram appears. This is achieved by calculating the respective probabilities *regardless* of the position of the n-gram in the message, by summing the individual counts for each n-gram position.

Pseudocode We provide a condensed pseudocode for both algorithms in Algorithm 1. In the case of the PMI$_{nc}$, the n-grams in the pseudocode would be whole messages, *i.e.*, trigrams. This base pseudocode would then be duplicated, interpreting the context as either an observation that may provide positional information (*e.g.*, *begin+1*) or an integer.

For the PMI$_c$ algorithm, only the unigrams and bigrams would be evaluated. The base pseudocode would also be duplicated, once for the integer in a given position, and second for the referent position. Each would be used as the context in which to evaluate the NPMI for each n-gram. A detailed commented pseudocode for both the PMI$_{nc}$ and PMI$_c$ algorithms is available in Algorithm 2 and Algorithm 3 in Appendix D, respectively.

Both algorithms use two hyperparameters: a confidence threshold t_c and top_n t_n. The confidence threshold refers to the value of the NPMI measure at which a message or n-gram can be considered to refer to the given part of the observation unambiguously. To account for polysemy (where one symbol can have multiple meanings), the agents can use a single n-gram to refer to multiple integers.

https://doi.org/10.52202/079017-4446

Algorithm 1: PMI Algorithm Base

Gather `ngram_counts`, `context_counts`, `joint_counts`, `n_grams`;

for *each n-gram g in position p and context c* **do**

$P(g,p) = \texttt{ngram\_counts}[g] \cdot \frac{1}{\texttt{total } n\texttt{-grams}}$;

$P(c) = \texttt{context\_counts}[c] \cdot \frac{1}{\texttt{total contexts}}$;

$P(g,p;c) = \texttt{joint\_counts}[(g,c)] \cdot \frac{1}{\texttt{total } n\texttt{-grams}}$;

$\text{NPMI}(g,p;c) = \log_2 \frac{P(g,p,c)}{P(g)P(c)} \cdot \frac{1}{-\log_2 P(g,p,c)}$;

end

return NPMI;

This is given by the second hyperparameter, top_n, which sets the degree of the polysemy, or the number of integers to be considered for a given n-gram.

5 Spatial referencing experiments

The agent pairs are trained over 16 different seeds to verify the results' significance. All agent pairs achieve above 98% accuracy on the referential task, showing that the agents develop a way to communicate about spatial relationships in their observations. The analysis provided in this section is based on the messages collected from the test dataset after the training has finished.

The two hyperparameters, t_c and t_n (Section 4), governing the NPMI measure have been determined through a grid search to maximise the understanding of the emergent language, by maximising the translation accuracy. The results in this section are obtained using the best-performing values for each of the hyperparameters. We provide the values for the grid search in Appendix A.

5.1 Emergence of non-compositional spatial references

Using the PMI_{nc} algorithm, we detect the emergence of messages tailored to convey the positional information contained in the observations. As mentioned in Section 2, sender observations which require shifting convey additional information about the position of the target within the sequence. In over 90% of agent pairs, these observations are assigned unique messages, used only for each kind of observation, *i.e.*, *begin*, *begin+1*, *end-1* and, *end*.

In 20% of runs which develop these specialised messages, the same repeating character is used to convey the message. The characters used for these observations are *reserved* only for these kinds of observations. For example, in one of the runs the agents use character 11 to signify the beginning of the sequence, with the character 11 being used only in two contexts: as the messages $[11, 11, 11]$ to signify *begin*, or as a message $[0, 11, 11]$ to signify *begin+1*. In other cases, characters are fully reserved for specific messages. *e.g.*, 22 is used only for *end*, in the message $[22, 22, 22]$.

The emergence of non-compositional references used for other observations is also detected using the PMI_{nc} algorithm. Such messages refer to a specific integer in a specific position of the sender observation, *e.g.*, $o_5 = 10$. While we allow for polysemy of the message in our analysis using $t\_n = [1, 2, 3, 5, 10, 15]$, we observe the highest translation accuracy with $t\_n = 1$, indicating that the non-compositional messages do not have any additional meanings.

5.2 Emergence of compositional spatial references

Using the PMI_c algorithm, we also detect the emergence of *compositional spatial references* for 25% of agent pairs. Such messages are composed of two parts, a positional reference and an integer reference. The positional reference specifies where a given integer can be found in the observation, in relation to the masked target integer -1. The integer reference specifies which integer the positional reference is referring to. For example, one pair of agents has assigned the unigram 7 to mean that the *target* integer is 2 to the right of the *given* integer, and the bigram $[0, 2]$ to mean the integer 18.

https://doi.org/10.52202/079017-4446

Table 1: Average emergence and vocabulary coverage of all message types.

Message Type	Avg. % Emergence	Avg. % of Messages
Non-Compositional Positional	99.3% (100%-93.75%)	1% (3%-0%)
Non-Compositional Positional Reserved	18.75% (18.75%-18.75%)	1% (3%-0%)
Non-Compositional Integer	45.1% (100%-0%)	10% (15%-0%)
Compositional Integer	100% (100%-100%)	34% (99.7%-0%)
Compositional Positional	25% (27%-0%)	56% (100%-0%)

Together, a message can be composed $[7, 0, 2]$, which means that the target integer for the receiver to identify is 2 to the right of the integer 18, *i.e.*, $\boldsymbol{o} = [18, X, -1, X, X]$. This allows the sender to identify the target integer exactly, given the sequence $\boldsymbol{s}$.

In Table 1, we summarise the emergence of each type of message across all runs, together with the percentage of the vocabulary that they represent. The entries in the table are composed of average percentages, across all t_n and t_c choices. In the parentheses, we show the maximum and minimum values across all t_n and t_c choices. The average % of emergence represents the absolute % of runs which developed that message type or message feature. For all messages, the average % of messages which are of a given type or exhibit a given feature is only counted for in runs where these features emerged.

5.3 Evaluating interpretation validity and accuracy

To ensure the validity of our message analysis, we present two hypotheses which, if supported by the results, would indicate that the mappings generated by the NPMI measure are correct.

Hypothesis 1 (H1) If the correlations exist and do not require non-trivial compositionality (Perkins, 2021), and are not highly context-dependent (Nikolaus, 2023), then the evaluation accuracy should be significantly higher than chance, or above 20%, when using the identified mappings.

Hypothesis 2 (H2) If the positional components of compositional messages are correctly identified and carry the intended meaning, then their inclusion should result in an increase in accuracy.

Given the messages identified by the NPMI method, we test **H1** and **H2** by using a dictionary of all messages successfully identified, given a value of both NPMI hyperparameters t_n and t_c. A dataset is generated to contain only targets which can be described with the messages present in the dictionary.

For the non-compositional messages, the dataset is generated by selecting a message from the dictionary at random, and creating an observation that can be described with that message. Given a non-compositional message that corresponds to the target being on the right of the integer 15, an observation $\boldsymbol{o} = [1, 15, -1, 5, 36]$ would be created. Analogously, for non-compositional positional messages such as *begin* an observation $\boldsymbol{o} = [-1, 15, 8, 5, 36]$ would be created.

For the compositional messages, we create the observations by randomly selecting a positional component and an integer component from the dictionary. For example, given the unigram 7 meaning that X is 2 to the left of the target, we could select the bigram $[8, 14]$ corresponding to the integer 30. The observation created could then be $\boldsymbol{o} = [30, 8, -1, 36, 5]$. The dataset creation process for the compositional messages also checks if the observations can be described given the two n-grams in their required positions within the message.

To test **H2**, a dataset is created using **only** the integers that can be described by the dictionaries, randomly selecting integer components from the dictionary, and creating the respective observations. This process also accounts for the required positions of the message components so that a message describing the observation can always be created. For example, if the unigram 9 described the integer 11, and the bigram $[5, 1]$ described the integer 6, a corresponding observation could be $\boldsymbol{o} = [11, 6, -1, 8, 9]$. The positions of the integers in the observations are chosen at random. By generating both compositional datasets using a stochastic process, we do not assume a specific syntax. Rather, the syntax can only be identified by looking at messages understood by the receiver.

Table 2: Accuracy improvements using the NPMI-based dictionary, $\pm$ denotes the 1-sigma standard deviation. Non-Compositional Positional refers to messages such as *begin* or *end*, Non-Compositional Integer refers to the non-compositional monolithic messages describing both the position and the integer, Compositional-NP refers to messages only containing the identified integer components, and the Compositional-P which refers to messages containing both the identified integer and positional components.

Dict Type	t_n	t_c	**Average Accuracy**	**Maximum Accuracy**
Non-Compositional Positional	1	0.9	90% $\pm$3%	**94%**
Non-Compositional Integer	1	0.5	36% $\pm$0.4%	37%
Compositional-NP	1	0.5	22% $\pm$ 2%	28%
Compositional-P	1	0.5[3]	30% $\pm$ 21%	78%

These datasets, together with their respective dictionaries, are then used to query the receiver agent, testing if the messages are identified correctly. We run this test for all of our trained agents, with the dictionaries that were identified for each agent pair. We provide the details in Table 2.

Using just the non-compositional positional messages, we observe a significant increase in the performance of the agents, compared to random chance accuracy of **20%**. This proves **H1**, showing that at least some messages do not require complex functions to be composed, or contextual information to be interpreted. As the accuracy for these messages reaches over 90% on average, we argue that the NPMI method has captured almost all the information transmitted using these messages.

As mentioned in **H2**, we examine the impact of the positional components and whether they carry the information the NPMI method has identified. We, therefore, separate the compositional analysis into two parts: Compositional-NP, where the positional components are replaced with 0, and Compositional-P, which includes the identified positional components. In the Compositional-NP case, the agents achieve a close to random accuracy, whereas, in the Compositional-P case, agents achieve above random accuracy, with some agent pairs reaching over 75% accuracy. This proves our **H2** correct, showing that the NPMI method has successfully identified the positional information contained in the messages, together with the integer information.

6 Discussion

Having successfully verified both **H1** and **H2**, we confirmed the validity of the language analysis.We also verify the generalisation ability of the agents, by evaluating varying training and evaluation sequence lengths, vocabulary sizes, and hidden size in Appendix C.

To provide human interpretability of the emergent language, we use the NPMI method to create a dictionary providing an understanding of both the positional and compositional messages. We present an excerpt from an example dictionary in Table 3. With human interpretability, we can gain a deeper understanding of the principles underlying the agents' communication protocol.

We posit that the emergence of compositional spatial references points to a first emergence of a simple syntactic structure in an emergent language. Both of the n-grams in our example from Section 5.2, also shown in Table 3, are assigned specific positions in the message by the agents. The unigram 7 must always be in the first position of the message, while the bigram $[0, 2]$ must always be in the second position. The emergence of this structure shows that even though referential games have been considered obsolete in recent research (Chaabouni et al., 2022; Rita et al., 2024), a careful design of the environment may yet elicit more of the fundamental properties of natural language.

We hypothesise that the emergence of non-compositional spatial references tailored to specific observations, such as *begin+1*, is due to observation sparsity. Compositionality would bring no benefit since the observations which they describe are usually rare, representing 1-2% of the dataset and are monolithic, *i.e.*, *begin*, *begin+1*, *end-1*, and *end*. We therefore argue that the emergence of non-compositional references in these cases is **advantageous**, since these messages are easily compressible. Since these messages are monolithic, they could be compressed to a single token/character in

[3] t_n for the referent position n-grams is set to 0.3

https://doi.org/10.52202/079017-4446

simple encoding schemes. In contrast, compositional messages require at least two tokens/characters, one for each integer/positional component. With a linguistic parsimony pressure (Chaabouni et al., 2019; Rita et al., 2020) applied, these messages could be more efficient at transmitting the information contained within these observations than compositional ones.

Table 3: Example dictionary of the agents' messages and their meanings

Message	**Type**	**Meaning**
$[11, 11, 11]$	Non-Compositional Positional	*begin*
$[0, 11, 11]$	Non-Compositional Positional	*begin+1*
$[10, 10, 10]$	Non-Compositional Positional	*end-1*
$[18, 18, 18]$	Non-Compositional Positional	*end*
$[12, 16, 14]$	Non-Compositional Integer	15 is 1 left of target
$[15, m_2, m_3]$	Compositional Positional	? is 2 left of target
$[7, m_2, m_3]$	Compositional Positional	? is 2 right of target
$[m_1, 0, 17]$	Compositional Integer	Integer 1
$[m_1, 0, 2]$	Compositional Integer	Integer 18
$[m_1, 8, 14]$	Compositional Integer	Integer 30

7 Limitations

The accuracy for the Non-Compositional Integer, and Compositional-P messages averages about 33%. While still above random, showing that some meaning is captured in non-compositional messages, it points to there being more to be understood about these messages. We hypothesise this may be due to the higher degree of message pragmatism, or context dependence (Nikolaus, 2023). Our method of message generation, using randomly selected parts, may not be able to capture the complexity of the messages. For example, the context in which they are used might be crucial for some n-grams, requiring the use of a specific n-gram instead of another when referring to certain integers, or when specific integers are present in the observation. Just like in English, certain verbs are only used with certain nouns, such as "pilot a plane" vs "pilot a car". While the word "pilot" in the broad sense refers to operating a vehicle, it is not used with cars specifically. This may also be the case for the emergent language. For compositional messages, an additional issue may be that some messages are non-trivially compositional, using functions apart from simple concatenation to convey compositional meaning (Perkins, 2021), making them impossible to analyse with the NPMI measure. However, these issues may be addressed by scaling the emergent communication experiments as the languages become more general with the increased complexity of their environment (Chaabouni et al., 2022).

8 Conclusion

Recent work in the field of emergent communication has advocated for better alignment of emergent languages with natural language (Boldt and Mortensen, 2024; Rita et al., 2024), such as through the investigation of deixis (Rita et al., 2024). Aligned to this approach, we provide a first reported emergent language containing *spatial references* (Lyons, 1977), together with a method to interpret the agents' messages in natural language. We show that agents can learn to communicate about spatial relationships with over 90% accuracy. We identify both compositional and non-compositional spatial referencing, showing that the agents use a mixture of both. We hypothesise why the agents choose non-compositional representations of observation types which are sparse in the dataset, arguing that this behaviour can be used to increase communicative efficiency. We show that, using the NPMI language analysis method, we can create a human interpretable dictionary, of the agents' own language. We confirm that our method of language interpretation is accurate, achieving over 94% accuracy for certain dictionaries.

Acknowledgments and Disclosure of Funding

This work was supported by the UK Research and Innovation Centre for Doctoral Training in Machine Intelligence for Nano-electronic Devices and Systems [EP/S024298/1].

The authors would like to thank Lloyd's Register Foundation for their support.

The authors acknowledge the use of the IRIDIS High-Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

Marco Baroni. Rat big, cat eaten! Ideas for a useful deep-agent protolanguage. *ArXiv preprint*, abs/2003.11922, 2020.

Brendon Boldt and David R. Mortensen. A Review of the Applications of Deep Learning-Based Emergent Communication. *Transactions on Machine Learning Research*, 2024.

Gerlof J. Bouma. Proc. of gscl. In *Von der Form zur Bedeutung: Texte automatisch verarbeiten - From Form to Meaning: Processing Texts Automatically*, volume 30, pages 31–40, 2009.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. Anti-efficient encoding in emergent communication. In *Proc. of NeurIPS*, pages 6290–6300, 2019.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. In *Proc. of ACL*, pages 4427–4442, 2020.

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 *Proc. of ICLR*, 2022.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*, pages 1724–1734, 2014.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proc. of ACL*, pages 76–83, 1989.

Roberto Dessì, Eugene Kharitonov, and Marco Baroni. Interpretable agent communication from scratch (with a generic visual processor emerging on the side). In *Proc. of NeurIPS*, pages 26937–26949, 2021.

Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi, and Yelena Yesha. Improving word similarity by augmenting PMI with estimates of word polysemy. *IEEE TKDE*, 25(6):1307–1322, 2013.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Proc. of NeurIPS*, pages 2149–2159, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proc. of ICLR*, 2017.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. EGG: a toolkit for research on emergence of lanGuage in games. In *Proc. of EMNLP*, pages 55–60, 2019.

Tomasz Korbak, Julian Zubek, and Joanna Raczaszek-Leonardi. Measuring non-trivial compositionality in emergent communication. In *4th Workshop on Emergent Communication, NeurIPS 2020*, 2020.

Angeliki Lazaridou and Marco Baroni. Emergent Multi-Agent Communication in the Deep Learning Era. *ArXiv preprint*, abs/2006.02419, 2020.

 https://doi.org/10.52202/079017-4446

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. In *Proc. of ICLR*, 2018.

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

Jia Peng Lim and Hady W. Lauw. Aligning Human and Computational Coherence Evaluations. *Computational Linguistics*, pages 1–58, 2024.

Olaf Lipinski, Adam J. Sobey, Federico Cerutti, and Timothy J. Norman. It's About Time: Temporal References in Emergent Communication. *ArXiv preprint*, abs/2310.06555, 2023.

John Lyons. Deixis, space and time. In *Semantics*, volume 2, pages 636–724. Cambridge University Press, 1977.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proc. of AAAI*, pages 1495–1502, 2018.

Jesse Mu and Noah D. Goodman. Emergent communication of generalizations. In *Proc. of NeurIPS*, pages 17994–18007, 2021.

Mitja Nikolaus. Emergent Communication with Conversational Repair. In *Proc. of ICLR*, 2023.

Denis Paperno and Marco Baroni. Squibs: When the whole is less than the sum of its parts: How composition affects PMI values in distributional semantic vectors. *Computational Linguistics*, 42 (2):345–350, 2016.

Hugh Perkins. Neural networks can understand compositional functions that humans do not, in the context of emergent communication. *ArXiv preprint*, abs/2103.04180, 2021.

Mathieu Rita, Rahma Chaabouni, and Emmanuel Dupoux. "LazImpa": Lazy and impatient neural agents learn to communicate efficiently. In *Proc. of CoNLL*, pages 335–343, 2020.

Mathieu Rita, Paul Michel, Rahma Chaabouni, Olivier Pietquin, Emmanuel Dupoux, and Florian Strub. Language Evolution with Deep Learning, 2024.

Shane Steinert-Threlkeld. Toward the Emergence of Nontrivial Compositionality. *Philosophy of Science*, 87(5):897–909, 2020.

Anton Thielmann, Arik Reuter, Quentin Seifert, Elisabeth Bergherr, and Benjamin Säfken. Topics in the Haystack: Enhancing Topic Quality through Corpus Expansion. *Computational Linguistics*, pages 1–37, 2024.

Ryo Ueda and Koki Washio. On the relationship between Zipf's law of abbreviation and interfering noise in emergent languages. In *Proc. of ACL*, pages 60–70, 2021.

Ryo Ueda, Taiga Ishii, and Yusuke Miyao. On the Word Boundaries of Emergent Languages Based on Harris's Articulation Scheme. In *Proc. of ICLR*, 2023.

Ryosuke Yamaki, Tadahiro Taniguchi, and Daichi Mochihashi. Holographic CCG Parsing. In *Proc. of ACL*, pages 262–276, 2023.

A Training Details

The computational resources needed to reproduce this work are shown in Table 4, with the hyperparameters in Table 5 and Table 6. The Table 4 shows resources required for all training and evaluation. The processors used were a mixture of Intel Xeon Silver 4216s and AMD EPYC 7502s. The GPUs used were a mixture of NVIDIA Quadro RTX 8000s, NVIDIA Tesla V100s, and NVIDIA A100s. These nodes used in our experiments were hosted on the IRIDIS cluster. The development process consumed more compute, which we estimate would have added 10 CPU and GPU hours, to account for experimentation.

Table 4: Compute resources

Resource	Value (1 Run)	Value (Training Total)	Value (Evaluation & Analysis)
Nodes	1	8	1
CPU	16 cores	128 cores	64 cores
GPU	1	8	1
Memory	50 GB	400 GB	120 GB
Storage	1 GB	32 GB	32 GB
Wall time	2 hours	240 hours	24 hours

Table 5: Hyperparameters

Parameter	Value
Epochs	1000
Optimizer	Adam
Learning Rate α	0.001
Gumbel-Softmax Temperature	[1.0]
Training Dataset Size	200k
Test Dataset Size	20k
No. Distractors	4
No. Points	[20,40,60,100]
Message Length	3
Vocabulary Size	[13,26,52]
Sender Hidden Size	[64,128]
Receiver Hidden Size	[64,128]

Table 6: PMI Grid Search Parameters

Parameter	Values
t_c	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
t_n	[1, 2, 3, 5, 10, 15]

B Dataset Details

To train and evaluate the agents, we use datasets consisting of 200,000 samples for training, 200,000 for validation, and 20,000 for testing. Each dataset is generated independently, with sequences created randomly. Given the sequence length of 60 and the fact that no integers are repeated, the number of possible permutations is $60! \approx 8 \times 10^{81}$, which vastly exceeds the number of samples we generate. We further ensure that there is no overlap between datasets by empirically checking the overlap rates across 1,000 randomly generated datasets, confirming an overlap rate of 0%.

C Generalisation

To generalise the results presented in this paper, we also run additional tests, varying the vocabulary size, training sequence length, evaluation sequence length, and the hidden size of the agents, as

 https://doi.org/10.52202/079017-4446

outlined in Appendix A. We observe no performance decline with either increasing or decreasing the vocabulary size or the training sequence length, given that the agents have enough capacity within their network to still learn the longer sequence lengths. We observe a decline in task accuracy at sequence lengths of 100, when the agents have a hidden size of 64. However, increasing the hidden size to 128 brings the training and validation accuracy back to over 90%.

When agents are evaluated on sequence lengths that are different from the ones they were trained on, we observe a small performance decline for small differences in sequence lengths. We present the average accuracies for the base case (Sequence shortened by 0), as well as the average difference in accuracy as compared to the baseline for different sequence lengths in Table 7. We observe a significant difference if the agents are evaluated on sequences that are over 50% shorter than the ones they were trained on. We hypothesise that this is due to the agents missing certain integers that they used more often than others, therefore reducing their accuracy. However, even in the worst case, the accuracy remains above 70%.

Table 7: Evaluation of different sequence lengths

Training sequence length	Sequence shortened by				
	0	-5	-10	-20	-40
20	98.68%	-0.53%	-7.50%	N/A	N/A
40	95.59%	0.05%	-0.75%	-5.55%	N/A
60	92.98%	0.30%	-0.30%	-2.47%	-15.8%
100	86.23%	0.34%	-0.03%	-1.26%	-5.2%

D Algorithm Descriptions

For our pseudocode we will be using the Python assignments convention, *i.e.*, $=$ and $\leftarrow$ are equivalent, and x+=1 is equivalent to $x \leftarrow x + 1$. The algorithms presented are for $top\_n = 1$. To improve the computational efficiency. the probability of the integer appearing is statically defined as $\frac{1}{60}$ for $top\_n = 1$, or in Equation (1) for $top\_n > 1$. In the case of $top\_n > 1$ we use the probability for the integer as per Equation (1), to account for the polysemy, *i.e.*, the probability for any of top_n integers occurring in the observation. The lower part of the binomial is 4, as there are 4 integers that can be sampled from the 60 possible integers, instead of 5, as we exclude the target integer.

$$p(integers) = \frac{\binom{60}{4} - \binom{60-top\_n}{4}}{\binom{60}{4}} \tag{1}$$

Additionally, in the PMI_c algorithm, we specify a probability to equal to 0.98 in Line 74 and Line 77. This is a simplification of the calculation for clarity of the pseudocode. This probability is instead obtained using the count of a given type of observation, divided by the number of total observations. This calculation is performed for each type of observation, *i.e.*, *begin*, *begin+1*, *end*, *end-1* and *middle*. The probability of the *middle* observation is very close to 1, being on average 0.98, while the other probabilities are on average 0.005. Since the *middle* observation is most common, we included its value in the pseudocode.

Algorithm 2: The PMI$_{nc}$ algorithm

Data: O_M ; # All observations together with sent messages
Data: $L = len(O\_M)$; # Total number of observations with sent messages
Data: $S = [begin, begin + 1, end - 1, end]$; # List of positional observations
Result: $pmi_{nc}[m][NPMI]$

pmi_{nc} = dict;
for $o, m \in O\_M$ **do**
  $pmi_{nc}[m][count]$ += 1 ; # Message occurrences
  for $pos \in S$ **do**
    if $o == pos$ **then**
      $pmi_{nc}[pos][count]$ += 1 ; # Positional observations count
      $pmi_{nc}[m][pos]$ += 1 ; # Message sent with positional observation
    end
  end
  for $integer \in o$ **do**
    $pmi_{nc}[m][integer\_pos][integer]$ += 1 ; # Message sent with integer in given position
  end
end
for $pos \in S$ **do**
  $posit_{total} = pmi_{nc}[pos][count]$; # Count of positional observations
  $p(pos) = \frac{posit_{total}}{L}$; # Estimate observation probability
  for $m \in pmi_{nc}[m]$ **do**
    $m_{total} = pmi_{nc}[m][count]$; # Total count of message
    $ms_{total} = pmi_{nc}[m][pos]$; # Total count of message with positional obs
    $p(m) = \frac{m_{total}}{L}$; # Estimate message probability
    $p(m, pos) = \frac{ms_{total}}{L}$; # Estimate joint probability
    $h(m, pos) = -\log_2(p(m, pos))$;
    $pmi(m, pos) = \log_2(\frac{p(m,pos)}{p(m)p(pos)})$;
    $npmi(m, pos) = \frac{pmi(m,pos)}{h(m,pos)}$;
    $pmi_{nc}[m][NPMI] = npmi(m, pos)$;
  end
end
for $pos \in pmi_{nc}[m]$ **do**
  for $integer \in pmi_{nc}[m][pos]$ **do**
    $p(pos) = \frac{1}{60}$; # Estimated observation probability for 60 integers
    $m_{total} = pmi_{nc}[m][count]$; # Total count of message
    $ms_{total} = pmi_{nc}[m][pos][integer]$; # Total count of message with integer in given position
    $p(m) = \frac{m_{total}}{L}$; # Estimate message probability
    $p(m, pos) = \frac{ms_{total}}{L}$; # Estimate joint probability
    $h(m, pos) = -\log_2(p(m, pos))$;
    $pmi(m, pos) = \log_2(\frac{p(m,pos)}{p(m)p(pos)})$;
    $npmi(m, pos) = \frac{pmi(m,pos)}{h(m,pos)}$;
    $pmi_{nc}[m][pos][integer][NPMI] = npmi(m, pos)$;
  end
end

https://doi.org/10.52202/079017-4446

Algorithm 3: The PMI$_c$ algorithm

Input: t_c ; # Confidence value
Data: O_M ; # All observations together with sent messages
Data: $L = len(O\_M)$; # Total number of observations with sent messages
Data: $ngrams$; # List of all message n-grams present in O_M
Result: $pmi_c[m][NPMI]$

$pmi_c = \text{dict}$;
; # First we identify n-grams corresponding to integers.
for $ngram \in ngrams$ **do**
  for $o, m \in O\_M$ **do**
    if $ngram \in m$ **then**
      $pmi_c[ngram][count]$ += 1 ; # Total n-gram occurrences
      $pmi_c[ngram][ngram\_pos][count]$ += 1 ; # n-gram occurrences including n-gram position
      for $integer \in o$ **do**
        $pmi_c[ngram][integer][count]$ += 1 ; # n-gram sent with integer in given position
        $pmi_c[ngram][ngram\_pos][integer][count]$ += 1 ; # n-gram in given position sent with integer in given position
      end
    end
  end
end
; # Calculate integer NPMI.
for $ngram \in ngrams$ **do**
  ; # Position variant NPMI.
  for $pos \in pmi_c[ngram][ngram\_pos]$ **do**
    $p(integer) = \frac{1}{60}$; # Estimated observation probability for 60 integers
    $integer_p = max(pmi_c[ngram][integer][count])$;; # Find integer with highest co-ocurrence given position
    $ngram_{pos} = pmi_c[ngram][ngram\_pos][count]$;
    $p(ngram_{pos}) = \frac{ngram_{pos}}{L}$
    $p(ngram_{pos}, integer) = \frac{pmi_c[ngram][ngram\_pos][integer][count]}{L}$;
    $h(ngram_{pos}, integer) = -\log_2(p(ngram_{pos}, integer))$;
    $pmi(ngram_{pos}, integer) = \log_2(\frac{p(ngram_{pos}, integer)}{p(ngram_{pos})p(integer)})$;
    $npmi(ngram_{pos}, integer) = \frac{pmi(ngram_{pos}, integer)}{h(ngram_{pos}, integer)}$;
    $pmi_c[ngram][ngram\_pos][integer] = npmi(ngram_{pos}, integer)$;
  end
  ; # Position invariant NPMI.
  $integer = max(pmi_c[ngram][integer][count])$;; # Find integer with highest co-ocurrence
  $p(integer) = \frac{1}{60}$; # Estimated observation probability for 60 integers
  $ngram_{total} = pmi_c[ngram][count]$;
  $p(ngram) = \frac{ngram_{total}}{L \times (4 - len(ngram))}$; # If n-gram is length 1, it could appear 3 times per message
  $p(ngram, integer) = \frac{pmi_c[ngram][integer][count]}{L}$;
  $h(ngram, integer) = -\log_2(p(ngram, integer))$;
  $pmi(ngram, integer) = \log_2(\frac{p(ngram, integer)}{p(ngram)p(integer)})$;
  $npmi(ngram, integer) = \frac{pmi(ngram, integer)}{h(ngram, integer)}$;
  $pmi_c[ngram][integer] = npmi(ngram, integer)$;
end

Algorithm 4: The PMI_c algorithm cont.

; # Now we identify n-grams corresponding to referent positions.
$ngram_{pr}$ = dict;
; # Prune n-grams with NPMI below c
for $ngram \in pmi_c$ **do**
  for $integer \in pmi_c[ngram]$ **do**
    if $pmi_c[ngram][integer] < t_c$ **then**
      del $pmi_c[ngram][integer]$;
    end
    for $pos \in pmi_c[ngram]$ **do**
      for $integer \in pmi_c[ngram][pos]$ **do**
        if $pmi_c[ngram][pos][integer] < t_c$ **then**
          del $pmi_c[ngram][pos][integer]$;
        end
      end
    end
  end
end
; # Find messages with integer n-grams
for $ngram \in pmi_c[ngram]$ **do**
  for $o, m \in O\_M$ **do**
    ; # Position variant n-gram
    if $pmi_c[ngram][pos]$ **then**
      if $ngram \in m[pos]$ **then**
        $new\_ngram = m - ngram$; # Get leftover n-gram
        $pr = pos(pmi_c[ngram][pos][integer], msg)$; # Get the possible referent position
        $ngram_{pr}[new\_ngram][pr][count] += 1$; # Count leftover n-gram occurence
        $ngram_{pr}[new\_ngram][pos][pr][count] += 1$; # Count leftover n-gram occurence in given positions
      end
    end
    ; # Position invariant n-gram
    else
      if $ngram \in m$ **then**
        $new\_ngram = m - ngram$; # Get leftover n-gram
        $pr = pos(pmi_c[ngram][integer], msg)$; # Get the possible referent position
        $ngram_{pr}[new\_ngram][pr][count] += 1$; # Count leftover n-gram occurence
        $ngram_{pr}[new\_ngram][pos][pr][count] += 1$; # Count leftover n-gram occurence in given positions
      end
    end
  end
end

https://doi.org/10.52202/079017-4446

Algorithm 5: The PMI_c algorithm cont.

; # Calculate referent position NPMI.

for $ngram \in ngram_{pr}$ **do**

  for $pr \in ngram_{pr}[ngram][pr]$ **do**

    ; # Position variant NPMI.

    for $pos \in ngram_{pr}[ngram][pos][pr]$ **do**

      $p(pr) = 0.98$; # Estimated observation probability for given position

      $ngram_{pos} = ngram_{pr}[ngram][pos][pr][count]$; $p(ngram_{pos}) = \frac{ngram_{pos}}{L}$

      $p(ngram_{pos}, pr) = \frac{ngram_{pr}[ngram][pos][pr][count]}{L}$;

      $h(ngram_{pos}, pr) = -\log_2(p(ngram_{pos}, integer))$;

      $pmi(ngram_{pos}, pr) = \log_2(\frac{p(ngram_{pos}, pr)}{p(ngram_{pos})p(pr)})$;

      $npmi(ngram_{pos}, pr) = \frac{pmi(ngram_{pos}, pr)}{h(ngram_{pos}, pr)}$;

      $pmi_c[ngram][pos][pr] = npmi(ngram_{pos}, pr)$;

    end

    ; # Position invariant NPMI.

    $p(pr) = 0.98$; # Estimated observation probability for given position

    $ngram = max(ngram_{pr}[ngram][pr][count])$; # Find highest positional reference count

    $p(ngram) = \frac{ngram}{L}$;

    $p(ngram, pr) = \frac{ngram_{pr}[ngram][pr][count]}{L}$;

    $h(ngram, pr) = -\log_2(p(ngram, integer))$;

    $pmi(ngram, pr) = \log_2(\frac{p(ngram, pr)}{p(ngram)p(pr)})$;

    $npmi(ngram, pr) = \frac{pmi(ngram, pr)}{h(ngram, pr)}$;

    $pmi_c[ngram][pr] = npmi(ngram, pr)$;

  end

end

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim that we demonstrate how agents can communicate about spatial relationships, and how such a language can be interpreted. These claims are supported by our results in Section 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A discussion of the limitations is provided in Section 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

https://doi.org/10.52202/079017-4446

Answer: [NA]

Justification: We present no theoretical results or proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture and training details are described in Section 3 and Appendix A respectively. The NPMI measures are described in Section 4, together with more detailed pseudocode in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code for the experiments is provided on GitHub (Footnote 2), together with the instructions on reproducing the paper's results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The training details are provided in Appendix A, with the optimisation method outlined in Section 3. Our code also includes detailed information about the training and test parameters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We include the value of the 1-sigma standard deviation for reported accuracies in Table 2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

 https://doi.org/10.52202/079017-4446

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All compute resources are specified in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have not identified any ethical concerns, regarding the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We do not believe that our methods for emergent language interpretability or the ability to use spatial references would have a path to significant negative societal impacts at this stage. We briefly discuss the positive impact of using spatial references and of more interpretable emergent languages in Section 1.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not expect increased emergent language efficiency or the ability to use spatial references to have a risk of misuse. We would argue more transparency into the emergent languages makes the systems less susceptible to misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The codebase used is credited with a citation in Section 3. The URL to the original code is also provided on GitHub. Both the original codebase, and our code for training and dataset creation, are released under the MIT Licence.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

 https://doi.org/10.52202/079017-4446

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code is well commented, and the documentation is available with the code on GitHub (Footnote 2) under the MIT Licence.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Human participants were not involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Human participants were not involved.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-4446