Connectionist Semantic Systematicity in Language Production

Jesús Calvillo (jesusc@coli.uni-saarland.de)
Harm Brouwer (brouwer@coli.uni-saarland.de)
Matthew W. Crocker (crocker@coli.uni-saarland.de)
Saarland University, Germany

Abstract
A novel connectionist model of sentence production is presented, which employs rich situation model representations originally proposed for modeling systematicity in comprehension (Frank et al., 2009). The high overall performance of our model demonstrates that such representations are not only suitable for comprehension, but also for modeling language production. Further, the model is able to produce novel encodings (active vs. passive) for a particular semantics, as well as generate such encodings for previously unseen situations, thus demonstrating both syntactic and semantic systematicity. Our results provide yet further evidence that such connectionist approaches can achieve systematicity, in production as well as comprehension.

Keywords: systematicity; sentence production; connectionist; semantics; syntax; neural networks

Introduction
A defining characteristic of human language is systematicity: “the ability to produce/understand some sentences is intrinsically connected to the ability to produce/understand certain others” (Fodor & Pylyshyn, 1988, p. 37). Further, Fodor and Pylyshyn (1988) argue that connectionist models are not able to display systematicity without implementing a classical symbol system.

The connectionist comprehension model developed by Frank et al. (2009), however, challenges this highly debated assertion, by developing a connectionist model of comprehension which is argued to achieve relevant levels of systematicity. Their model constructs a situation model (see Zwaan and Radvansky (1998)) of the state-of-affairs described by a sentence that also incorporates world knowledge-driven inferences. When the model processes a sentence like ‘a boy plays soccer’, for instance, it not only recovers the explicit, literal propositional content, but also constructs a more complete situation model in which a boy is likely playing outside on a field, with a ball, with others, and so forth. In this way it differs from other connectionist models of language comprehension and production, that typically employ simpler meaning representations, such as case-roles (Chang, Dell, & Bock, 2006; Mayberry, Crocker, & Knoeerle, 2009; Brouwer, 2014, among others). Crucially, Frank et al. (2009)’s model generalizes to both sentences and situations that it has not seen during training, exhibiting different levels of semantic systematicity and is argued to provide an important step in the direction of psychologically plausible models of language comprehension.

In the present paper, we examine whether the approach developed by Frank et al. (2009), is equally well suited to language production, and present a connectionist production model that generates sentences from these rich situation models. We show that our model successfully learns to produce sentences from these rich meaning representations. Crucially, we demonstrate that this model is able to describe unseen situations, demonstrating semantic systematicity similar to Frank et al. (2009), as well as produce alternative encodings (e.g. active/passive) for a given situation, that were not seen during training and thus demonstrate syntactic systematicity.

Method
We employ an extended Simple Recurrent Neural Network architecture (SRN) (Elman, 1990) to generate sentences from rich semantic representations, called Distributed Situation Space (DSS) vectors. The DSS model (Frank, Koppen, Nordman, & Vonk, 2003; Frank et al., 2009) is a distributed scheme for meaning, in which the meaning of a situation—a state of affairs—is represented as a situation vector in a high-dimensional “situation-state space”.

This section is organized as follows: first, we introduce the Distributed Situation Space as described by Frank et al. (2003, 2009); then, we explain the vectors that we use as well as the architecture of the model; afterwards the training and evaluation schema is presented; and finally we present the results obtained from the evaluation.

Distributed Situation Space
The DSS model defines a microworld in terms of a finite set of basic events (e.g., play(charlie, chess) and place(heidi, bedroom))—the smallest meaning-discerning units of propositional meaning in that world. While these basic events can be defined in several ways, we adopt the microworld presented by Frank et al. (2009). Situations in a microworld are represented in terms of these basic events, which can be conjoined to form complex events (e.g., play(charlie, chess) ∧ win(charlie)). Not all conjunctions of basic events are, however, possible or equally likely. That is, world knowledge can pose both hard (e.g., physical) and probabilistic (e.g., preferential) constraints on event co-occurrence. A situation-state space is a large set of \( m \) situations defined in terms of \( n \) basic events, effectively yielding an \( m \times n \) matrix (see Table 1). Each of the \( m \) situations in this matrix is encoded by setting basic events that are the case in a given situation to 1 (True) and those that are not to 0 (False). A situation-state space matrix is constructed by sampling \( m \) situations (using a non-deterministic sampling proce-
The vectors that we use for production are slightly differ-
ence two of their paper), consisting of 44 basic events, cen-
dimensional situation vectors to 150-
dimensional representation plus an extra bit indi-
corporating whether the system must produce an active sentence (1), or a passive one (0). The dimensionality of the output layer matches the number of available words in the grammar (42), plus the end-of-sentence marker.

Time in the model is discrete. At each time-step \( t \), activation propagates forward following the trajectory: input \( \rightarrow \) hidden \( \rightarrow \) output. The activation of the output layer yields a probability distribution over the available words. We define the word produced at time-step \( t \) as the one with highest probability (highest activation). The model stops producing words when an end-of-sentence marker has been produced.

The hidden layer also receives a copy of its own activation pattern at the previous time-step \( t - 1 \), through a 120-unit context layer, which is set to zero at the beginning of each sentence. These units help to preserve some memory of what has been produced expanding several time steps in the past, and allow the model to generate sentences of variable length.

In addition, the hidden layer receives the identity of the word that was produced at time-step \( t - 1 \) (zeros at \( t = 0 \)) through monitoring units that connect the output layer to the hidden layer, where only the output unit corresponding to the produced word at time-step \( t - 1 \) is activated (set to 1), while all other units are set to 0.

Finally, the hidden and output layers also receive input from a bias unit (with a constant activation value of 1).

The Model

Our model architecture, as we can see in Figure 1, is broadly similar to the one used by Frank et al. (2009), with the main difference being that the inputs and outputs are reversed; it maps DSS representations onto sequences of localist word representations. Similar to Frank et al. (2009), we stress that it is not intended to model human language development.

The model consists of a 45-unit input layer, a 120-unit recurrent hidden (tanh) layer, and a 43 unit (softmax) output layer. The dimensionality of the input layer corresponds to the 44 basic events in the microworld, plus one extra binary unit that indicates whether the model must output an active sentence (1), or a passive one (0). The dimensionality of the output layer matches the number of available words in the grammar (42), plus the end-of-sentence marker.

The examples set consists of a set of pairs \( \{ (DSS_1, \phi_1), \ldots, (DSS_n, \phi_n) \} \) where each \( DSS_i \in \{ 0, 1 \}^{45} \) corresponds to a DSS representation plus an extra bit indicating whether the system must produce an active sentence (1) or a passive one (0); and \( \phi_i = \{ sent_1, \ldots, sent_k \} \) where \( sent_j \) is a sentence, a sequence of words \( word_1, \ldots, word_n \), expressing the information contained in \( DSS_i \). Each set \( \phi_i \) represents all the possible sentences that express the
information contained in the corresponding DSS, and in the expected voice.

The sentences that we use are those generated by the microlanguage defined by Frank et al. (2009) (see their Tables 5–8). This microlanguage consists of 40 words that can be combined into 13556 sentences according to its grammar. We minimally modified this grammar. First, we introduced the determiners that were missing (a, the); and second, we added an end-of-sentence marker (a period) to the sentences. Leaving a total of 43 words that were encoded at the output layer as localist vectors.

Sentences that expressed unlawful situations according to the microworld rules, and therefore whose DSS belief vectors were empty, were discarded; leaving a total of 8201 lawful sentences. From these, 6782 sentences were in active voice and 1419 in passive. Note that this set contains sentences with simple semantics (e.g., “charlie plays chess.” → voice and 1419 in passive. From these, 6782 sentences were in active 

The first set (setAP) allowed for three different testing conditions: active sentences, and a passive is queried.

Condition 1: Situations for which the model has seen only active sentences, and a passive is queried.

Condition 2: Situations for which the model has seen only passive sentences, and an active is queried.

Condition 3: Completely new situations (not seen during training), passive and active sentences are queried.

The second set (setA) allowed for two different testing conditions:

Condition 4: Situations for which the model has seen only active sentences, and a passive is queried.

Condition 5: Completely new situations (not seen during training), passive and active sentences are queried.

These conditions represent different levels of generalization or systematicity. In all cases, the queried sentence type has never been seen by the model. For conditions 1, 2 and 4 the model has seen the situation but not in the queried voice. Importantly, for conditions 3 and 5, the model has never seen the situation itself. Finally, for conditions 4 and 5, where passives are queried, not only the system has not seen the passive sentences, but also they are not defined by the grammar.

SetAP was randomly shuffled and split into 10 folds of 90% training and 10% testing DSS representations, meaning per fold 382 training and 42 testing situations. For each fold, the testing DSS representations were further split into the 3 conditions, rendering 14 different testing DSS representations per condition, per fold.

SetA was also shuffled and split into 10 folds, but in order to preserve uniformity, for each fold and for testing, 14 DSS representations were drawn per condition; meaning that each fold contained 28 testing and 330 training DSS representations.

Finally, for condition 1 the DSS representations were coupled with their corresponding active sentences and incorporated into the training set (while the passive sentences remained in the testing set); vice-versa for condition 2. Similarly, for condition 4 the active sentences were incorporated into the training set, while during testing the system will be queried for a passive construction, even though there is none according to the grammar.

Training Procedure We trained the model using cross-entropy backpropagation (Rumelhart, Hinton, & Williams, 1986) with weight updates after each word in the sentence of each training item. Prior to training, all weights on the projections between layers were initialized with random values drawn from a normal distribution \( \mathcal{N}(0, 0.1) \). The weights on the bias projections were initially set to zero.

During training, the monitoring units were set at time \( t \) to zero for \( t = 0 \). This reflects the notion that during training the word contained in the training sentence at time-step \( t - 1 \) should be the one informing the next time step, regardless of

Training and Evaluation

In order to assess the performance of the model in terms of accuracy and generalization, we employed a 10-fold cross-validation schema. First, we divided the DSS representations into two sets: the first one (setAP) corresponding to those associated to both active and passive sentences and the second one (setA) corresponding to DSS representations that were related only to active sentences.

The first set (setAP) allowed for three different testing conditions:

- Condition 1: Situations for which the model has seen only active sentences, and a passive is queried.
- Condition 2: Situations for which the model has seen only passive sentences, and an active is queried.
- Condition 3: Completely new situations (not seen during training), passive and active sentences are queried.

The second set (setA) allowed for two different testing conditions:

- Condition 4: Situations for which the model has seen only active sentences, and a passive is queried.
- Condition 5: Completely new situations (not seen during training), passive and active sentences are queried.

These conditions represent different levels of generalization or systematicity. In all cases, the queried sentence type has never been seen by the model. For conditions 1, 2 and 4 the model has seen the situation but not in the queried voice. Importantly, for conditions 3 and 5, the model has never seen the situation itself. Finally, for conditions 4 and 5, where passives are queried, not only the system has not seen the passive sentences, but also they are not defined by the grammar.

SetAP was randomly shuffled and split into 10 folds of 90% training and 10% testing DSS representations, meaning per fold 382 training and 42 testing situations. For each fold, the testing DSS representations were further split into the 3 conditions, rendering 14 different testing DSS representations per condition, per fold.

SetA was also shuffled and split into 10 folds, but in order to preserve uniformity, for each fold and for testing, 14 DSS representations were drawn per condition; meaning that each fold contained 28 testing and 330 training DSS representations.

Finally, for condition 1 the DSS representations were coupled with their corresponding active sentences and incorporated into the training set (while the passive sentences remained in the testing set); vice-versa for condition 2. Similarly, for condition 4 the active sentences were incorporated into the training set, while during testing the system will be queried for a passive construction, even though there is none according to the grammar.

Training Procedure We trained the model using cross-entropy backpropagation (Rumelhart, Hinton, & Williams, 1986) with weight updates after each word in the sentence of each training item. Prior to training, all weights on the projections between layers were initialized with random values drawn from a normal distribution \( \mathcal{N}(0, 0.1) \). The weights on the bias projections were initially set to zero.

During training, the monitoring units were set at time \( t \) to zero for \( t = 0 \). This reflects the notion that during training the word contained in the training sentence at time-step \( t - 1 \) should be the one informing the next time step, regardless of
the previously produced (and possibly different) word. During testing, the monitoring units are set to 1.0 for the word that was actually produced and 0.0 everywhere else.

The model was trained for a maximum of 200 epochs, each epoch consisting of a full presentation of the training set, which was randomized before each epoch. Note that each item of this set consisted of a DSS, paired with one of the possible sentence realizations describing the state of affairs represented in DSS. Hence, during each epoch, the model saw all the possible realizations of DSS contained in the training set for a given fold. We employed an initial learning rate of 0.124 which was halved each time there was no improvement of performance on the training set during 15 epochs. No momentum was used. Training halted if the maximum number of epochs was reached or if there was no performance improvement on the training set over a 40-epoch interval.

Sentence Level Evaluation For a given DSS representation DSS, the model produces a sequence of words ŝi constituting a sentence describing the state-of-affairs represented in DSS. Because a DSS can be described by one or more sentence(s), we assume that the output of the model is perfect if the sentence produced ŝi is part of the set φi of all possible realizations of DSS in the queried voice.

However, sometimes the output of the model ŝi for a DSS does not perfectly match any of the sentences in φi. As such, we also compute the similarity between the output of the model ŝi and each sentence in φi. This similarity is derived from their Levenshtein distance (Levenshtein, 1966); which is the number of insertions, deletions or substitutions that are needed in order to transform one string into another. More formally, Levenshtein similarity sim(s₁, s₂) between two sentences s₁ and s₂ is defined as:

$$sim(s_1, s_2) = 1 - \frac{distance(s_1, s_2)}{\max(length(s_1), length(s_2))}$$  \hspace{1cm} (1)

where distance is the Levenshtein distance. This similarity measure is 0 when the sentences are completely different and 1 when they are the same. Thus, for each item i in the training and test set, we obtain a similarity value:

$$sim(\hat{s}_i) = \max_{s \in \phi_i} sim(\hat{s}_i, s)$$  \hspace{1cm} (2)

Results

We trained 10 instances of the model, corresponding to each fold as described above. Each instance was initialized with a different set of weight matrices. The scores reported below are averaged over these instances.

On the training set, the model achieved an average similarity score of 99.43% (and 98.23% perfect matches). This shows that the model is able to learn to transform a DSS situation vector into a sequence of words describing the state-of-affairs that the vector represents.

Regarding the test conditions, Table 2 shows the average similarity scores for each of them. For conditions 4 and 5, where passives are queried but there are no example sentences given by the grammar, no similarity scores can be computed and in exchange a qualitative analysis will be presented.

We can notice a slight drop of similarity scores for condition 5. This could be explained because setA in general contained fewer sentences per DSS, and thus fewer training items.

The average similarity score across all conditions is of 97.1%, with 88.57% of perfect matches. This translates into roughly one to three mistakes per condition and per fold. The nature of these mistakes is addressed in the next section, however we can observe that the performance in terms of similarity is very high and almost perfect in several cases.

Qualitative Analysis Although the performance is quite high, the model elicits a number of systematic mistakes that provides us with some insight into the internal mechanism that is implemented by the network. Examples of these are shown in Table 3.

Taking a qualitative look into the produced sentences, it is evident that, with literally a couple of exceptions, all the sentences produced are syntactically correct and semantically felicitous. The vast majority of the elicited mistakes occur when the model produces a sentence that is semantically highly similar to the one expected. This pattern can be seen already during training, where the mistakes correspond to situations that are closely related, so the model is unable to distinguish them, even though it has already seen the situations/sentences (examples 1-3 in Table 3).

For the conditions shown in Table 2, the errors elicited during 5 folds were manually inspected in order to see their regularities. The errors observed (38 in total) can be classified into 2 main categories: the first one (63.2%) being errors concerning over and underspecification, and the second one (31.6%) corresponding to situations that because of the design of the network the model are remarkably similar, differing only in one aspect.

Concerning the first category, the errors can be further split into location under- (21.05%) and over- (7.9%) specification (examples 4-5 in Table 3), subject under- (15.8%) and over- (10.5%) specification (examples 6-7 in Table 3), and object under- (2.6%) and over- (5.2%) specification (examples 8-9 in Table 3).

The errors contained in the second category (examples 10-12 in Table 3) correspond to sentences that at first glance seem correct, it is only after taking a deep look into the micro-world that one can see the mistake. First, according to this

<table>
<thead>
<tr>
<th>Cond.</th>
<th>Query</th>
<th>Similarity (%)</th>
<th>PerfectMatch (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pas</td>
<td>97.66</td>
<td>87.86</td>
</tr>
<tr>
<td>2</td>
<td>act</td>
<td>97.58</td>
<td>92.86</td>
</tr>
<tr>
<td>3</td>
<td>act</td>
<td>98.35</td>
<td>93.57</td>
</tr>
<tr>
<td>3</td>
<td>pas</td>
<td>96.79</td>
<td>83.57</td>
</tr>
<tr>
<td>5</td>
<td>act</td>
<td>95.08</td>
<td>85.0</td>
</tr>
</tbody>
</table>
microworld, whenever there is a winner, there is also a loser, which means that sentences that are apparently contradictory ("someone loses." vs "someone wins.") actually have the same implications within the microworld and therefore are semantically identical. Second, in general whenever there is a winner/loser, the loser is usually situated in the same location as the winner. However, only for the game hide_and_seek and when the participants are inside, the loser can be in the bedroom, while the winner could be in the bathroom, or vice versa. Finally, whenever there is a prepositional phrase ("in the bedroom"), it is attached to the subject of the sentence according to the grammar, which means that in “Heidi beats Sophie in the bedroom”, Heidi is in the bedroom while Sophie could be in either the bedroom or the bathroom, while in “Sophie loses to Heidi in the bedroom”, it is Sophie who stays in the bedroom while Heidi could also be in the bathroom. Apart from this detail, the situations are almost identical.

According to the errors so far analyzed, we can conclude that the nature of these is primarily related to situations that are highly similar.

With regard to the test conditions 4 and 5 where a passive sentence is queried but the grammar does not properly define its characterization, Table 4 presents examples of output sentences and the situations that they were supposed to portray. As mentioned before, these situations can be of two types: the first one involving a winning/losing situation where both actors are explicitly mentioned, and the second type being situations where the object of the action is not defined. We took a closer look into the output of the model for these conditions by manually making an analysis of the output of 3 folds (84 situations), whose results we will now present.

Even though in condition 4 the model has not seen the type of sentences that are queried, and that in condition 5 the model has no experience with the queried situations, the sentences produced by the model are mostly correct and coherent with the semantic information that is given to it. One can see that some information is omitted, however, this is expected since the grammar itself does not contain rules that allow to fully encode these situations.

In general, the model learns during training that passive sentences always start by mentioning the object of the action, and that this object is never a person. Therefore, for each situation it tries first to find this object and then it tries to describe the rest of the situation.

Concerning winning/losing situations (examples 1-2 in Table 4), which conform 92.9% of the analyzed situations, the object is always a game because in the microworld winning/losing can only happen while playing a game. Thus, the model produces the specific name of the game when it is known (e.g. "soccer is..."), otherwise the sentence starts with "a game is...". Then one of the players is mentioned (one omitted) and the rest of the situation is portrayed.

For the case of situations with an underspecified object (7.1%, examples 3-4 in Table 4), it is unknown whether the subject is playing a game or with a toy, so the model is forced to choose one. In all cases the model chooses a toy, which seems reasonable because mostly the DSS representations of the underspecified sentences are more similar to situations where a toy is played with. For example, the DSS representation of “someone plays.” is more similar to the one of “someone plays with a toy.” (99.36% cosine similarity) than to the representation of “someone plays a game.” (97.73% cosine similarity).

Similar to the other conditions, errors regarding over and underspecification occur in conditions 4 and 5, but are rare (example 5 in Table 4). And finally one type of error that appears only for these situations corresponds to the interchange of the winner/loser in game situations (example 6 in Table 4).

In sum, one can see from the output that the model is able to take the linguistic elements learned during training in order to characterize situations for which it has no experience, while being as informative as possible. The only difficulty appears to be the distinction of situations that are highly similar. However, the performance of the model is very high in general and even for the sentences that do present a mistake, the output is largely correct.

Discussion
The model learns to generate sentences from rich situation representations, and furthermore, it generalizes to novel sentences and situations.

We can see that the model is able to learn the syntactic
The boy loses to Heidi at chess in the bedroom.
a girl beats Charlie with difficulty in the street.
chess is lost by Heidi in the bedroom.
a toy is played with.
Sophia beats Charlie with ease.
a game is lost with difficulty by Charlie.
a toy is played with in the playground by Sophia.
a game is won with ease by Sophia.

words to its vocabulary.
was our main objective. In principle, the microworld can be
levels of systematicity for unknown areas of this space, which
that we use is finite, however we were able to show different
associate passive constructions with these situations.
mar that was used to construct the training/testing sets does
demonstrated in test conditions 3 and 5, since the semantic
they are related.
Crucially, the model also generalizes semantically, as
demonstrated in test conditions 3 and 5, since the semantic
representations given to the model are completely novel to
it, so any correct output can be regarded as arising from the
regularities within the microworld from which the DSS rep-
resentations are derived—cf. the comprehension results by
Frank et al. (2009).

We hypothesize that the fact that the model has difficul-
ties with highly similar situations means that the model is
able to roughly reconstruct the topography of the microworld
semantic space, putting together situations that are semanti-
cally related. At the same time, the model assigns linguistic
structures to each area in this semantic space such that se-
mantically similar situations are assigned linguistically simi-
lar realizations. Given that in practice the semantic space is a
continuous 44-dimensional space, in theory the model should
be able to generate sentences for unseen areas as long as it is
given enough information during training in order to recon-
struct the semantic space and the mapping between semantics
and linguistic realizations, as proposed by Frank et al. (2009).

The results of the test conditions show that it is indeed the
case. Conditions 3 and 5 demonstrated that the model is able
to generate sentences for unseen areas in the semantic space,
therefore showing semantic systematicity. Conditions 1 and
2 demonstrated that the model is able to generate sentences
for semantically known situations but with a different voice
(active/passive), showing syntactic systematicity. Conditions
4 and 5, where passive sentences are queried, demonstrated
that it is able to produce coherent sentences even if the gram-
mar that was used to construct the training/testing sets does
not associate passive constructions with these situations.

As it is now, the semantic space related to the microworld
that we use is finite, however we were able to show different
levels of systematicity for unknown areas of this space, which
was our main objective. In principle, the microworld can be
extended by adding elements to the set of basic events, and
words to its vocabulary.

<table>
<thead>
<tr>
<th>Output</th>
<th>Active Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hide_and_seek is won with ease by Heidi in the playground.</td>
<td>Heidi beats Sophia with ease in the playground at hide_and_seek.</td>
</tr>
<tr>
<td>2 a game is won with ease by Sophia.</td>
<td>Sophia beats Charlie with ease.</td>
</tr>
<tr>
<td>3 a toy is played with.</td>
<td>someone plays.</td>
</tr>
<tr>
<td>4 a toy is played with in the playground by Sophia.</td>
<td>Sophia plays in the playground.</td>
</tr>
<tr>
<td>5 a game is lost with difficulty by Charlie.</td>
<td>a girl beats Charlie with difficulty in the street.</td>
</tr>
<tr>
<td>6 chess is lost by Heidi in the bedroom.</td>
<td>the boy loses to Heidi at chess in the bedroom.</td>
</tr>
</tbody>
</table>

Conclusion

We presented a novel connectionist model of sentence pro-
duction that uses the rich semantic representations described
in Frank et al. (2009). This model is able to take such repre-
sentations and produce sentences that accurately describe the
associated situations. The model is also able to produce alter-
native encodings (e.g. active vs. passive) for a particular
semantics, showing an overall high performance and demonstr-
ating that these semantic representations are not only suit-
able for comprehension, but also for modeling language pro-
duction. And furthermore, the model is able to generate novel
sentences for previously unseen situations, thus demonstrat-
ing syntactic and semantic systematicity.

References

Brouwer, H. (2014). The electrophysiology of language com-
prehension: A neurocomputational model. Unpublished
doctoral dissertation, University of Groningen.
Psychological review, 113(2), 234.
Science, 14(2), 179–211.
cognitive architecture: A critical analysis. Cognition, 28(1-
Connectionist semantic systematicity. Cognition, 110(3),
358–379.
Frank, S. L., Koppen, M., Noordman, L. G. M., & Vonk,
Levenshtein, V. I. (1966). Binary codes capable of correct-
ing deletions, insertions, and reversals. In Soviet physics
Learning to attend: A connectionist model of situated lan-
Learning representations by back-propagating errors. Na-
ture, 323(6088), 533–536.
in language comprehension and memory. Psychological