Learning complex linguistic structure:
The Simple Recurrent Network as a model of nonadjacent dependency learning

Jon Willits
University of Wisconsin Madison

Abstract
Higher-order syntactic structure is a hallmark of language, contributing to its expressive power, but also making it more difficult to learn. One such example is the nonadjacent dependency, where one element predicts another element, but at a distance. The present work reviews behavioral studies of nonadjacent dependency learning in artificial grammars, focusing on a few critical studies that highlight important properties of language. In this work, we explain behavioral performance in those studies within a single theoretical framework, the connectionist simple recurrent network (SRN). Five simulations were performed, showing that the SRN captures the qualitative patterns of human performance in these learning tasks. Further, examination of why the SRN succeeds provides additional insights into how people may learn language. One important conclusion is that the SRN, like people, performs dramatically better in situations where the learning situation is made more realistic, incorporating cues like meaning and variability that children have when learning natural languages.
1. Introduction

There are many properties that make human language unique and powerful. For example, language contains complex patterns and regularities at many levels, giving rise to an expressive power unmatched by other communicative systems. Some of these patterns can be characterized by local predictive relationships, such as the co-occurrence probabilities amongst phonemes, syllables, and words. Across development, humans are exquisitely sensitive to these adjacent regularities, suggesting that this form of learning plays an important role in language acquisition (for a review, see Romberg & Saffran, 2010).

Natural languages also contain higher-order structure that transcends simple associations between adjacent elements. Nonadjacent dependencies – in which one unit in language predicts a second unit, but at a distance – exist in all human languages, and at virtually every level of linguistic structure. There are nonadjacent dependencies (also sometimes called long-distance dependencies) between low-level units like phonemes – such as how nonadjacent sequences of consonants form word roots in many languages (such as English irregulars like *drink, drank, drunk*; and the consonants that form noun frames in Semitic languages). There are also nonadjacent dependencies between higher-level units, such as number agreement between subject nouns and verb phrases in English (e.g., *the cat sleeps* vs. *the cats sleep*), which can be separated by embedded clauses (e.g., *the cats that I saw yesterday sleep in the shed*). Nonadjacent dependencies in grammatical structure convey critical information. They also vastly increase the complexity of language and the information it contains.

Explaining how people learn and represent knowledge of these dependencies has been a major endeavor. To some, the sheer difficulty of learning and representing this information has been a major argument in favor of rule-based and syntactically structured representations of language (Chomsky, 1957, 1959). However, two subfields of research provide evidence that this leap is unnecessary, and that nonadjacent dependencies are quite compatible with statistical learning and association-based perspective on language acquisition.

The first subfield is research using artificial grammars to attempt to study novel nonadjacent dependency learning. This work initially suggested that nonadjacent dependencies are very difficult to learn, perhaps providing addition evidence against a statistical learning-based account. However, a number of more recent studies have made an interesting point: the only situations where people have an extreme difficulty learning nonadjacent dependencies are when all other possible cues to learning the nonadjacency have been stripped away, leaving the nonadjacent dependency itself as the only thing that can be learned. This is, of course, extremely divorced from the situations in which these dependencies occur in natural language. In natural language, the nonadjacent dependencies that need to be learned come with a number of other structural, perceptual, or semantic cues that make them easier to learn. And in experiments that provide learners with access to these cues find that learning nonadjacent dependencies proceeds much more easily in these cases (Gómez, 2002; Newport & Aslin, 2004; Willits, Lany, & Saffran, 2012).

The second subfield of research that supports the viability of statistical learning-based accounts of nonadjacent dependencies in language is research on the behavior of recurrent connectionist networks, such as the Simple Recurrent Network (Elman, 1990, 1991; Cleeremans & McClelland, 1991; Allen & Seidenberg, 1999; Christiansen & Chater, 1999). This research has shown that these types of networks can learn to represent surprisingly complex sequential dependencies. And in a nice correspondence with the behavioral research, it has been found that
connectionist networks also learn structure better in situations that show create correspondence to the actual conditions of human language learning.

The following work has three main goals. The first goal is to review research on human learning of nonadjacent dependencies, with a specific focus on a few topics that are considered by many to be critical aspects of language’s complexity and expressive power. Here, we will show that some of the most important issues related to nonadjacent dependency learning can be captured using a single theoretical framework, the Simple Recurrent Network (SRN). The second goal will be to show that, not only does the SRN successfully model an important range of phenomena, it also provides critical insights into the learning processes that we wouldn’t have had if we hadn’t used this approach. Finally, the third goal will be using the correspondence between the behavioral data and the SRN model to reinforce the broader point that the ecological validity of these experiments – retaining the cues that people are likely using to learn the relationships – is critical in formulating adequate theories of human language learning and representation.

1.1. Nonadjacent dependency knowledge in children and adults

Given the importance and ubiquity of nonadjacent dependencies in language, it is perhaps not surprising that infants and children are sensitive to native-language dependencies at young ages. For example, 18-month-old infants discriminate between sentences based on the grammaticality of a match between a verb auxiliary and a verb inflection (e.g., *is cooking* versus *can cooking*; Santelmann & Jusczyk, 1998). Likewise, in sentences like “Chris has a red ball but Max doesn’t have one,” 17-month-olds consistently expect the anaphor one to refer to red ball, not simply to ball (Lidz, Waxman, & Friedman, 2003). By 3 to 5 years of age, children can correctly link pronouns to nouns within complex sentences and larger units of discourse (Arnold, Brown-Schmidt, & Trueswell, 2007). More generally, children learn nonadjacent dependencies involving complex semantic, pragmatic, and discourse structure in their native language, such as the relationship between alternative syntactic structures (e.g., active sentences versus passive sentences) and different meanings (e.g., Gertner, Fisher, & Eisengart, 2006, Naigles, Gleitman, & Gleitman, 1992), and also how to use prior contexts to correctly comprehend ambiguous input, even when these prior contexts do not immediately precede the ambiguous element (e.g., Booth, Harasaki, & Burman, 2006; Khanna & Boland, 2010; Simpson & Foster, 1986).

Adults are sensitive to an even larger range of nonadjacent dependencies involving phonology, morphology, syntax, and semantics (MacDonald & Seidenberg, 2006). Furthermore, experienced language users are sensitive to both the abstract and item-specific dimensions that relate nonadjacent dependencies. For example, adults have general knowledge about verbs’ abstract argument structures, such as the number and types of nouns that are required by each verb. In addition to this abstract knowledge, adults also have expectations about specific verbs’ typical arguments, such as that the verb *give* is likely to have an animate agent as its subject and an inanimate direct object (Hare, McRae, & Elman, 2003; Tanenhaus et al., 1989), or that *pasta* takes a singular verb while *noodles* takes a plural one (Bock et al., 2001; Haskell, Thornton, & MacDonald, 2010).

1.2. Attempts to demonstrate learning of novel nonadjacent dependencies
Given these facts about the nonadjacent structure present in natural languages, and the typical ability to master it by adulthood, there is an intriguing disconnect that merits explication. On the one hand, nonadjacent dependencies are ubiquitous in natural language, and people clearly learn them in everyday life. However, in the laboratory, new nonadjacent dependencies seem to be very difficult to learn. A number of studies using artificial grammar learning paradigms have assessed the conditions under which adults and infants can learn nonadjacent relationships. As described below, these studies have found that nonadjacent dependencies either take an extremely long time to learn (in some cases taking thousands of trials over a week’s worth of experimental sessions, Cleeremans & McClelland, 1991), or else they require special circumstances in order for the learning to take place in a more reasonable amount of time. This raises a puzzle: if nonadjacent dependencies are so ubiquitous, and people do learn them, why is it so difficult to demonstrate this learning in the laboratory? The difficulty in demonstrating learning of novel and arbitrary nonadjacent dependencies is especially interesting in light of the historical debate about the sufficiency (or lack thereof) of association-based learning mechanisms for acquiring and representing knowledge of higher-order linguistic structure (e.g., Chomsky, 1957; Elman, 2009).

The most common paradigm used to study nonadjacent dependency learning involves familiarizing participants with strings of nonsense words that contain nonadjacent dependencies, followed by a test phase. For example, Gómez (2002) familiarized adults and 18-month-old infants with words fitting an AxB grammar, where a particular A-element perfectly predicted which B-element would occur after an intervening x-word. Infants heard the following six strings during a familiarization phase (randomized, for two minutes): pel wadim rud, pel kicey rud, pel puser rud, vot wadim jic, vot kicey jic, and vot puser jic. These strings contained two nonadjacent dependencies (pel-x-rud and vot-x-jic). Following familiarization, the test contrasted strings that followed the nonadjacent dependency (pel kicey rud) with strings that violated the dependency (pel kicey jic). Gómez found that infants and adults failed to learn this extremely simple dependency when given two to five minutes of exposure to the test corpus. In addition to these studies by Gómez, the literature demonstrates numerous failures to learn similar nonadjacent dependencies in short laboratory settings (e.g., Cleeremans & McClelland, 1991; Johnson et al., 2009; Lany, Gómez, & Gerken, 2007; Gómez & Maye, 2005; Misyak & Christiansen, 2007; Newport & Aslin, 2004; Pena et al., 1999).

However, most of these studies also revealed that participants have an easier time learning new nonadjacent dependencies under specific circumstances. The pattern of successes and failures is instructive about the process of language learning and language representation, as situations where learning is easier can provide insight into the mechanisms being used. Gómez (2002) noted one such example of where infants and adults can more easily learn a nonadjacent dependency. Gómez found that if the set size of the intervening words (e.g., the x-items in Gómez’s AxB grammar) is considerably higher than the set size of the nonadjacent pair, this makes the nonadjacent dependency much easier to learn. This situation is analogous to cases of natural language nonadjacencies like “is-x-ing”, where the number of x-items that occur in the “is-x-ing” frame (e.g., verbs) is very large. In addition, both adults and infants learn a new nonadjacent dependency more readily if they have previous experience with adjacent dependencies between words that are later encountered in parallel nonadjacent dependencies (e.g., Cleeremans & McClelland, 1991; Lany, Gómez, & Gerken, 2007; Lany & Gómez, 2008; Servan-Schreiber, Cleeremans, & McClelland, 1991). This is analogous to the process of learning about noun phrases in natural languages. Experience with determiner-noun relationships...
may facilitate subsequent learning of determiner-noun relationships that are separated by adjectives. Infants are also able to learn nonadjacent structure in linguistic and nonlinguistic sequences, as long as the nonadjacent structure involves repetition of specific elements in the sequence, such as an ABA pattern (Johnson et al., 2009; Marcus et al. 1999; Saffran et al., 2007). Nonadjacent relations (including the repetition studies above) can also be more easily learned if the nonadjacent elements are at the beginning and end of the sequence rather than embedded within a sequence (Endress, Scholl, & Mehler, 2005; Pena et al., 1999).

Newport and Aslin (2004) discovered another condition under which adults can more easily learn a nonadjacent dependency in a segmentation task. Consistent with the findings of Gómez (2002), adults failed to learn dependencies between nonadjacent syllables (e.g., ba-pe-di, ba-ru-di, ba-to-di). However, they successfully learned nonadjacent dependencies between consonants separated by vowels (e.g., hapedi, hepodu, bupoda). Newport and Aslin (2004) suggest that a Gestalt-like perceptual grouping process (Wertheimer, 1938) leads learners to process the perceptually similar consonants independently of the vowels. This allows learners to track nonadjacent structure within the perceptually grouped element sequences. Other studies using non-linguistic stimuli reveal a similar pattern: adjacent and nonadjacent dependencies in sequences are more readily tracked when the consistently co-occurring elements share perceptual features (Bower, 1972; Creel, Newport, & Aslin, 2005; Fiser & Aslin, 2002; Gerbhart, Newport, & Aslin, 2009; Turk-Browne et al., 2008).

There is a common theme in all of these studies that show circumstances where nonadjacent dependencies can be learned. In each situation, the situation is being made more ecologically valid, more like the actual situation in which language learners actually find themselves when trying to learn natural languages. Natural languages have a large number of cues (such as perceptual similarity and differences in variability) that can be exploited by a learner to find and represent structure. There is an additional cue that has not been studied in previously that is also highly correlated with many nonadjacent structures in language: meaning.

1.3. Using Semantic Knowledge to Learn About Linguistic Structure

Most high-level nonadjacent dependencies in natural languages (such as those between verbs and their noun arguments) have semantic correlates that predict the nonadjacent dependency. For example, it is not an accident that ditransitive verbs like give typically co-occur with three nouns (an agent, a patient, and a recipient); the meaning of give requires these three relations. Likewise, the types of nouns that typically fill these roles are not random or unpredictable. The semantics of the verbs and nouns, and our knowledge about the objects and events to which they refer, lead to reliable co-occurrence patterns both in language and in our experiences of real-world events. Language learners may be sensitive to these correlations between semantic or conceptual relations and linguistic structure, and may be able to use their knowledge about semantic relations to bootstrap learning about nonadjacent linguistic structure. Indeed, recent studies investigating adult processing of combinatorial dependencies have found that the presence of semantic information plays a critical role in processing and acquisition of natural and artificial languages (Amato & MacDonald, 2010; Gillespie & Pearlmutter, 2011; Wonnacott, Newport, & Tanenhaus, 2008). Therefore, if we expand Newport and Aslin’s “perceptual grouping” hypothesis to become a more general “similarity grouping” hypothesis, including semantically or conceptually correlated cues in addition to perceptually correlated cues, the potential explanatory power of this hypothesis grows dramatically.
Several recent studies have confirmed that correlated semantic cues help adults and children learn novel nonadjacent dependencies (Willits, Lany, & Saffran, 2012; Willits & Saffran, 2012). In the first experiment, 24-month-old children were given a two-minute familiarization containing three-word sequences following an AxB grammar, where the first word (A) in each sequence perfectly predicted the third word (B) in each sequence. Children in the “Semantic Match” condition heard strings where matched A and B items in each string were from the same taxonomic category (“cookie-X-banana” and “doggy-X-kitty”). Children in the “Semantic Mismatch” condition heard strings where the A and B items were from opposite categories (“cookie-X-kitty” and “doggy-X-banana”). In a test phase children from both conditions were played all four types of strings, to see if they discriminated the sequences they heard during familiarization from sequences they hadn’t heard. Children in the semantic match condition showed evidence of discrimination, whereas infants in the semantic mismatch condition did not.

Follow-up experiments showed that the knowledge nonadjacent structure that children learned via these correlated semantic cues was not limited to simple string memorization; children were able to transfer this knowledge in a number of different ways. The first follow-up experiment showed that children transfer this knowledge of nonadjacent structure learned via correlated semantic cues to novel strings (at test, distinguishing strings which maintain the A-B contingencies from those that do not, even when different intervening x-items were used during familiarization versus at test). A second follow-up experiment showed that infants who are given a different words in familiarization and test, but words that maintain the category-nonadjacency relationship (e.g. trained on strings like “cracker-X-apple” and “bird-X-fishy” and then tested on the same strings as in the initial experiment) continue to show discrimination in the Semantic Match condition, but not in the Semantic Mismatch condition.

These experiments show that very young children can learn nonadjacent structure in language when there is a correlated semantic cue; specifically, when the words participating in the nonadjacency are similar to each other, they can be bound into knowledge about a nonadjacent dependency. This suggests that there is indeed a more general principle at work behind the perceptual-similarity bootstrapping proposal by Newport and Aslin (2004), and that perhaps any kind of similarity detectable by the cognitive system may be useful for learning about higher-order linguistic structure.

And there are indeed many ways in which the words that participate in nonadjacent dependencies that can be thought of as similar. Content nouns are more similar (both distributionally and referentially) to each other than they are to the function words that tend to break them up. Likewise, content words participating in nonadjacencies tend to share a number of semantic attributes, such as number and other semantic properties that get bound to them in language via inflections and case marking. In addition, the words that participate in specific semantic relations (such as teacher, taught, and student, or pilot, flew, and plane) have been shown to be distributionally similar to each other; because these words (or their referents) tend to participate in the same events, they are then similar in terms of sharing a set of co-occurring contexts, such as other words or other referential contexts that also tend to participate in those events (Willits et al., 2007).

However, one might question just how much nonadjacency knowledge this similarity-bootstrapping mechanism can explain. There are, of course, many ways that the elements in nonadjacencies are different, as well. Many of the different noun arguments for verbs (such as agents, patients, locations, instruments, recipients), have no shared semantic features, and often vary in a number of properties (such as animacy) that are known to be very important
determinants of conceptual similarity. What characteristically defines most thematic role structures in language is that there is a regular semantic relation between the roles: the verb *give* tends to involve (though is not required to involve) a relation between two animate participants and a third, concrete object which is getting transferred. The verb *give* is, by definition, the arbitrary label that happens to define this particular relation between this particular set of participants. Thus, the key to many nonadjacent relations in language is that they define an arbitrary but (relatively) consistent set of relations between particular participants, where the relation itself is the semantic relation that binds the participants.

Willits and Saffran (2012) tested whether adults could learn novel nonadjacent dependencies that contained these “arbitrary but consistent” semantic relations, using an AxB grammar with four nonadjacent pairs and four possible intervening items. In a two-minute familiarization phase, adult participants in a “Semantic Match” condition heard exemplars of the four nonadjacent pairs “doggy-X-kitty”, “bird-X-fishie”, “cookie-X-banana”, and “toast-X-cheese”.. The first and third items were always from the same taxonomic category. In contrast, participants in a “No Semantic Contingency” condition heard the pairs “doggy-X-kitty”, “bird-X-cheese”, “cookie-X-banana”, and “toast-X-bird”; half the time the words were in the same taxonomic category, but half the time they were opposite categories, resulting in no consistent semantic cue to the nonadjacent dependency. In a test phase, participants were given Familiarization-Consistent and Familiarization-Inconsistent strings (though which ones were consistent and inconsistent depended on their familiarization condition) and asked to identify the strings they had seen during familiarization. Replicating the effect with infants, participants in the Semantic Match condition were significantly more accurate at identifying which string they heard during familiarization, whereas participants in the “No Semantic Contingency” condition where at chance. Critically, a third “Semantic Mismatch” condition was also run, where participants’ familiarization was the four strings “doggy-X-banana”, “bird-X-cheese”, “cookie-X-kitty”, and “toast-X-fishy”. The first and third words were always from opposite taxonomies, an arbitrary but consistent semantic relation. The word pairs in each nonadjacent dependency were not similar to each other, but they were consistent in how they were paired. In the test phase, participants in this Semantic Mismatch condition were also significantly above chance at identifying the correct strings they heard during familiarization, and were in fact as accurate as those participants in the Semantic Match condition. Thus, Willits and Saffran (2012) demonstrate that adults’ ability to learn novel nonadjacent dependencies is quite robust as long as there is any sort of a consistent semantic mapping between the nonadjacent dependency and the semantic structure of the participants in the relation.

1.4. General Mechanisms for Learning Higher-order Linguistic Structure

The studies described above all suggest that nonadjacent dependencies are more easily learnable when the learning situation contains some of the cues that people actually have when learning natural language. But even with this realization, there are still unanswered questions about the exact learning mechanisms. For example, Gómez (2002), Newport and Aslin (2004), and Willits, Lany and Saffran (2012) identify factors that might “guide attention to the nonadjacent structure” or “provide a correlated cue to the nonadjacent structure.” But what, exactly, does that mean? It would be useful to have a mechanistic account of the learning and knowledge representation process by which these cues guide learning.
Equally important, it would be useful to have description of specific mechanisms that would allow us to distinguish between more rule-based accounts of linguistic structure knowledge, and more item-based or association-based accounts. Human behavior descriptively exhibits behaviors that are like both of these phenomena. If one were going to provide a general account of higher-order linguistic knowledge, it would need to capture both the ways in which language has rule-like generalizations, and the ways in which language is sensitive specific associations and distributional details. A general theory of higher-order linguistic structure must explain how many nonadjacent relationships in language seem to exist as symbolic relations between participants, such as verbs and their arguments. Verbs like give can have a non-animate recipient and an abstract patient (“the man gave the school an idea”), but the extent to which a particular set of arguments violates the probabilistic expectations of a particular construction or relation, will have consequences for how that particular combination is learned and used. Can both types of behaviors be explained by a single account, or must we posit both types of mechanisms in order to explain both of these behaviors?

One potential account of higher-order structure that may be able to capture both types of effects within a single mechanism is a Parallel Distributed Processing (PDP) model (Rumelhart & McClelland, 1986), the simple recurrent network (Elman, 1990). Simple recurrent networks (SRNs) are specifically designed to capture the structure of event sequences by combining information about the current input with information about the contextual state (e.g. what the model has experience in previous time steps).

A number of researchers have explored the ability of recurrent networks to model a number of behavioral phenomena. Cleeremans and McClelland (1991) found that SRNs are quite good at modeling learning event sequences. SRNs or more complex recurrent models have been shown to be good models of word segmentation (Christiansen, Allen, and Seidenberg, 1998; Elman, 1990), sentence comprehension (Allen & Seidenberg, 1999; Elman, 1991; Tabor and Tanenhaus, 1998), language production (Dell, Chang, & Griffin, 1999), phonemic and orthographic sequences, and the mapping between the two (Plaut et al., 1996; Seidenberg & McClelland, 1989), and memory for serial order (Botvinick and Plaut, 2006).

There are, however, a number of arguments for why recurrent networks should not be able to account for higher-level linguistic knowledge. Most commonly, it is argued that recurrent network models are not capable of learning abstract, symbolically-structured knowledge of the kind that many researchers think must exist, in order to explain a number of language phenomena (Fodor & Pylyshyn, 1988; Marcus, 1998a, 1998b; Pinker & Prince, 1988). One of the most important of those phenomena is nonadjacent structure in language, in the many different places it manifests itself in language at high and low levels. For the most part, these arguments about the fundamental capabilities of recurrent networks (both pro and con) take the form of abstract claims about the inherent properties of these networks, without strong tests or thorough knowledge of what they can actually do (though see Marcus, 1998a and 1998b, and Christiansen & Chater, 1999). Therefore, a thorough test and understanding of the capabilities of recurrent networks with regard to nonadjacent dependency learning is crucial.

A number of studies have begun investigating how recurrent networks handle nonadjacent structure. Cleeremans and McClelland (1991) trained human participants on sequences of events that followed a complex artificial grammar containing nonadjacent dependencies, and found that people had difficulty learning the nonadjacent relations. However, they did find that with much exposure, participants could learn the nonadjacent relations that were separated by up to three intervening items. Cleeremans and McClelland than modeled this behavior using an SRN,
finding that the SRN behaved in a qualitatively similar way, able to learn the same complex artificial grammar with much exposure and being sensitive to the distance between the nonadjacent elements.

Elman (1991) trained a simple recurrent network (SRN) on simple sentences containing a number of nonadjacent dependencies like number agreement (e.g. “boy§ who like girls chase§ girl” vs. “boy who likes girls chase§ girl”). Elman found that SRNs could perform considerably above chance at these nonadjacent relations, but also that there were a number of interesting ways in which the networks failed to learn nonadjacent structure. In an intriguing follow-up, Howell, Becker, and Jankowicz (2005) tested whether or not an SRN was better at learning word sequences when the model had access to semantic information. To test this, Howell et al. used distributed representations as input, (e.g. each input to the model was a vector of many values, with the similarity of those vectors representing whether or not the words were semantically similar). Howell et al. found that the SRN’s ability to predict words marginally improved when semantically structured distributed vectors were used. Howell et al. did not, however, specifically investigate how or whether this network improved performance on nonadjacent relations.

Christiansen and Chater (1999) performed the most thorough exploration of recurrent networks’ ability to learn nonadjacent structure. The trained SRNs on sentences that contained a number of different recursive structures, such counting recursive sequences (a sequence that has a particular number of A-elements, followed by an identical number of B-elements), center-embedded recursive sequences, cross dependency recursive sequences (sequences like ABAB, where the A-elements predict the A-elements and the B-elements predict the B-elements), and right branching recursive sequences. Christiansen and Chater studied how various factors about the input (like the distance separating the nonadjacently related items) and the network architecture (such as the number of units in the hidden layer) affected the network’s ability to learn these structures. They found that in most cases, the networks were able to learn these nonadjacent structures. More importantly, Christiansen and Chater found that the network’s pattern of successes and failures at learning these structures corresponded well to the sequences with which people have difficulty learning and processing these sequences (see also Christiansen & MacDonald, 2009).

The research done this far on SRNs and their ability to learn nonadjacent dependencies is suggestive but not decisive. At least two outstanding questions remain. First, given the recent research demonstrating that people learn nonadjacent dependencies better under important circumstances that are more ecologically valid, it would be extremely useful to know whether or not SRNs also demonstrate this characteristic, and whether or not an SRNs performance in these situations would provide into the learning mechanisms in these situations. Second, a number of researchers remain very skeptical about the SRNs utility as a model of human language learning, due to arguments about SRNs inability to learn and represent true abstract, rule-like relations. Clear demonstrations of SRNs abilities (or lack therefor) in this regard would be quite valuable.

1.5. Proposed Studies

The goal of the present work is threefold. The first is to establish whether or not the SRN is a plausible model of a number of critical nonadjacent dependency learning phenomena. By looking at how SRNs perform across all of these tasks, whether they succeed at learning a nonadjacent dependency, and especially understanding why they succeed or fail, we can take a step closer to having a unified account of learning nonadjacent structure, across the many
circumstances that make it easy and difficult. Will SRNs succeed in mimicking the human data and all circumstances, and do so for similar reasons, providing a somewhat singular explanation for these disparate phenomena? Or will SRNs success or lack thereof be more idiosyncratic from circumstance to circumstance? Though these analyses we will try to fulfill a second goal: does the SRN actually provide insight into nonadjacent dependency learning, above and beyond merely fitting the data?

The final goal is to reinforce the point that perhaps the reason that people have so much difficulty learning nonadjacent dependencies in artificial grammars is because those grammars tend to strip away all of the cues that people can actually use to solve the learning problem. If you add perceptual or semantic similarity back into the task, the task gets easier. If you add variability or noise back into the learning situation, often this actually makes the learning easier, not more difficult. We will explore this issue in simple recurrent networks, to see if they demonstrate the same effect? Does making the input into an SRN more realistic make the learning problem easier, or more difficult?

2. Simulation Overview

As described above, research to date has suggested that SRNs may be a good model of how people learn nonadjacent dependencies, but there are a number of serious open questions with regard to this proposal. Recent behavioral work has found a number of situations in which this learning gets easier, and it is an open question whether SRNs will also demonstrate this behavior in each of these cases, and help explain why, and thus lead to a useful comprehensive theory. In addition, there are a number of unresolved questions about the SRN’s ability to learn and represent abstract structure and rule-like knowledge, of the type that appears to be ubiquitous in language.

In the following five simulations, we explore these two main issues. Across all the simulations, the general task will be the same: the model will be given a sequence of strings containing nonadjacent dependencies, and the model’s task will be to learn the sequence by making predictions about what which item should come next. The model receives feedback about whether that prediction was correct, and adapts its knowledge representations based on that feedback. Below, I describe the general modeling framework to be used to simulate a number of different nonadjacency learning phenomena, and how the model will be assessed. Then, in the following sections, I describe the specific models and their performance.

Simulation 1 will involve teaching an SRN to learn a simple AxB grammar, and also tests whether the model can transfer what it learns about a particular nonadjacent dependency to novel strings (as people can do, Gomez & Maye, 2005; Willits, Lany, & Saffran, 2012).

Simulation 2 will investigate how an SRN performs at learning a nonadjacent dependency of different lengths, whether or not the SRN can transfer knowledge about a nonadjacent dependency of one length to different length, and whether or not SRNs can learn distance-invariant nonadjacent dependencies.

Simulation 3 will investigate whether the variability between nonadjacent items (i.e. whether having a small vs. large set of items that come in between nonadjacently related items) affects how SRNs learn nonadjacent dependencies in the same way as people (Gomez, 2002, Onnis et al., 2012).

Simulation 4 will explore how similarity relations amongst nonadjacently related items affects learning, and whether or not shared similarity and “arbitrary but consistent” relationships
affect the SRN the same way that they affect human learning (Newport & Aslin, 2004; Willits, Lany, & Saffran, 2012; Willits & Saffran, 2012).

Simulation 5 will explore the extent to which purely localist SRNs (i.e., SRNs without any microfeatures on which to assist learning) are capable of learning abstract, rule-like relations, as infants and adults are capable of doing (Marcus et al., 1999, Saffran et al., 2007).

2.1. Model Details

2.1.1. Core Features. The goal of this work is to provide a unified account of as wide a range as possible of nonadjacent learning phenomena. Each model will vary slightly to capture important differences between the phenomena or tasks being simulated; however, a number of critical features will stay the same across each situation that is being modeled.

A. Neural Network Implementation. The model used will be a simple connectionist network composed of sets of interconnected units, with weights specifying how strongly each unit is connected to each other unit. The units in the model will be divided into an input group, used to specify a representation of the each input stimulus in each sequence; an output group, used to specify a representation of each output response (which will also be the network’s prediction about which unit should come after the current input stimulus in the sequence); and a hidden group that mediates between the input and output groups (See Figure 1).

B. Recurrent Connectivity. A critical property of the model will be its recurrent connectivity. This recurrent connectivity will allow the model to feedback information about its own previous states in ways that will be critical to it forming internal representations of sequential structure. In the model, the hidden layer will feedback into itself via a copy function, as well as receive input via recurrent connections from the output layer. In other words, for any given input, the activation level of the hidden layer will be determined by a combination of three factors: (i) activation coming from the input layer; (ii) activation feeding back from the hidden layer’s activation state at for the previous input; (iii) and activation feeding back from the output layer’s activation state for the previous input.

![Figure 1. Architecture of the network used in all simulations.](image-url)
C. *Weight-based Encoding*. Knowledge and learning in the model are encoded in terms of the weighted connections between units, and how the weights for these connections change as a function of learning. The goal of the network will be to learn a set of weights such that, for any given input, the model’s weights lead to activation in the output layer that is a prediction of the input layer for the next input in the sequence. The model can therefore be thought of as learning to predict the next element in a sequence, based on what stimulus is the current input. Critically, for the model to be successful, it will have to learn one set of weights that will work for all sequences in each training corpus. Over the course of doing this, the model will attempt to find a set of weights into the hidden layer such that the hidden layer’s activation for each input can be thought of as comprising a representation of each environmental state: what the current input is (the weights into the hidden layer from the input layer), and what the current environmental context is (the weights coming into the hidden layer from the previous state’s hidden layer and output layer). The combination of these weights comprises the networks knowledge of how to represent each input correctly. Finally the weights from the hidden layer to the output layer can be thought of as a representation of how the model should respond to its representation of the current environmental state (i.e. what prediction to make about what the next output should be).

2.1.2 *Model Architecture*. In all the following simulations, near-identical networks were used, with the exception of the first simulation where a number of network parameters were systematically explored. In all the simulations, the network was composed of simple units divided into an input layer, an output layer, and a hidden layer. For all of the simulations, the size of the input and output layers varied (each unit is used to solely and uniquely represent a different stimulus), containing between 10 and 36 units depending on the number of stimuli in each simulation. The size of the hidden layer also varied, containing between two and 44 units. In some simulations the size of the hidden layer is fixed; in others it varied parametrically in order to study the effect of different hidden layer sizes on learning.

The activation level of each unit varied between zero and one and was set according to a standard approach (Rumelhart & McClelland, 1986); each unit’s activation level was calculated as a function of its current net input:

\[
\text{net}_j = \sum_i a_i w_{ij},
\]

(1)

Where \(a_i\) is the activation of each input unit \(i\) and \(w_{ij}\) is the weight of connection from unit \(i\) to unit \(j\). For units in the hidden layer, final activation levels were determined using a logistic function of the net input (using a gain parameter setting of 1 for all simulations):

\[
a_i = 1 / (1 + e^{-\text{net}_i})
\]

(2)

In simulations where the network was learning to turn on one (and only one) output unit, activations in the output layer were based on the softmax function.
\[ p_t = \frac{\exp(q_i)}{\sum_{j=1}^{n} \exp(q_j)}. \]  

(3)

Where \( p \) is the value of the target output node, and \( q \) and \( i \) index all the units in the output layer. The goal of the softmax function is to simulate competition between each unit in the layer, resulting in the activation of all units in the output layer summing to one. When using the softmax activation function, these output activation levels can be thought of as the network representing the posterior probability of each output (Rumelhart et al., 1995). In simulations where the network was learning to turn on multiple output units, activations in the output layer were based on the same sigmoidal function in (2).

The network’s connections were shown in Figure 1. Each unit also received input from a single bias unit, with a fixed activation of one. The weights were initialized to random values between -0.5 and 0.5 (except in the first simulation, where the effect of different weight initializations was explored). The network operated in discrete time steps, with each step corresponding to the presentation of a new input, activation flowing through the network, and the network making a prediction about what the next input should be at the output layer.

### 2.1.3. Task and Representation

In each of the localist simulations described below, the network’s task was to take an input comprised of sequences following an grammar such that in each sequence there is one item that is perfectly predicted by a previously occurring item. All of the grammars will follow some form of an AxB pattern, where the particular A-item that occurs in slot one perfectly predicts which B-item will occur; critically, the x-items that come between the A and B items will vary randomly and be of no predictive value as to which B should occur. In the purely localist simulations (sections 2.2, 2.3, and 2.4), a different input and output unit was used for each distinct A, x, and B stimulus in both the input and output layers. Thus, at each input, one and only one input unit was activated, and the output layer represented the network’s guesses of the posterior probabilities of the likelihood each stimulus coming next in the sequence. To simulate the course of an entire experiment, the model was presented with each item in the training corpus, one at a time, just as human participants are.

### 2.1.4. Training

The model was presented with an input, it made a prediction in its output layer, and then this prediction was compared to the target output. Divergence error was calculated across each unit

\[ \sum_j t_j \log \left( \frac{t_j}{a_j} \right), \]  

(4)

where \( j \) indexes each output unit. This measure of divergence error was then used to train the model using a version of recurrent backpropogation through time (Williams & Zipser, 1995). The learning rate (how much each weight was changed as a function of its contribute to error) was set at 0.001 and momentum was used for all simulations (except in the first simulation, where the effect of different learning rates and use of momentum were explored). Weights were updated after each event; each time the network received an input (\( A_i \) or \( x_i \)), its activation was propagated through the network, the output activation was compared to the target activation for that event, and the weights were adjusted as a function of the error on that item.

For each simulation, 30 different models were trained, starting from different randomly initialized weights. Each model was trained until it reached a predetermined level of overall...
error, corresponding to optimal prediction performance in the task (or as close as the model could get to optimum performance). This corresponded to the model learning that it was receiving AxB sequences, and thus learning when to expect an x-item (following A-items), when to expect B-items (following x-items), and which particular B-item to expect (depending on which A item had occurred previously). The model would never be able to predict exactly which x-item it should see, as the probability of these was always equally distributed; but the model could learn that the units corresponding to these two stimuli are the only ones it should activate. In other words, if there were four x-items, then when an A-item is the input, the model should learn to activate each x-item to a level of 0.25 (the probability of each item occurring), and all other units’ activations should be zero. In contrast to the x-items, the model’s input will result in a deterministic presentation of each B-item, contingent on the previous nonadjacent A-item. If the model has learned the optimum representation for the training corpus, when each x-item input is activated, it should learn to activate the correct B-item in the output layer based on the previous A-item (to a level of 1.0), and all other activations should be set to zero.

2.1.5. Evaluation of Performance. Each simulation was evaluated in two ways. First, did the model achieve criterion performance, learning to probability match in the cases where no perfect prediction could be made (A- and x-items), and learning to predict the correct B-item for each sequence? Second, what was the rate (in terms of number of iterations through the training corpus) in which the model learned each training corpus, or achieved its asymptotically lowest level of error if it did not fully learn? The second criterion is critically important for comparing the models’ performance to human data. As noted, in some cases people have great difficulty learning nonadjacent sequential structure; in other cases they find it considerably easier. Given enough exposure, SRNs will be able to learn any sequence it is given (at least, any of the relatively easy sequences it will be given in the following studies). Similarly, given enough exposure, presumably any person would likely learn the sequences as well (as demonstrated by Cleeremans & McClelland, 1991). The critical test, then, becomes comparing the relative difficulty that the SRN has with learning various sequences, and whether or not the qualitative pattern of difficult resembles the difficulty that humans demonstrate.

3. Simulation 1: Simple AxB Learning

The first simulation will have the simple goal of simulating learning of the simplest possible AxB grammar, and analyzing the effects of manipulating several of the network’s parameters, such as number of hidden units and the learning rate. This first simulation had two goals: (1) to achieve a qualitative understanding of the basic nature of how an SRN represents and learns nonadjacent dependencies, and (2) to establish a baseline of the paradigm’s performance at learning nonadjacent structure in the simplest case. While the network in this first simulation is in some ways the simplest representation of the problem, it is also the situation in which humans find learning the dependency most difficult. Thus, the speed with which the network learns the structure in simulation can serve as a benchmark against which later networks can be judged.

3.1. Training Corpus, Test Items, and Design

The design for this simulation was deliberately minimal, and is shown in Table 1. The training corpus had two nonadjacent pairs (A1→B1 and A2→B2), and three possible intervening
x-items (x1, x2, x3), for a total of six possible AxB strings. During training, the network was presented with each string as an individual sequence. For example, the network was first presented with the sequence “A1-x1-B1”, then it was presented with “A2-x3-B2”, and so on. The network experienced all six sequences in a blocked fashion, with the sequence order randomized within each block.

For testing, the network was presented with 12 strings: the six on which it had been trained, and also six that were inconsistent with training and that violated the nonadjacent dependency (e.g. A1-x1-B2). In addition, the network was also presented with strings that contained a novel intervening item (A1-x4-B1 and A1-x4-B2) in order to test whether the network was merely learning to memorize each triplet, or was in fact learning a more abstract and transferable fact about the A1-B1 relation. In behavioral experiments, both infants and adults trained on a set of nonadjacent dependencies clearly show the ability to transfer that knowledge to novel strings with new intervening items (Gómez & Maye, 2005; Willits, Lany, & Saffran, 2012). Thus, it is important that an SRN be able to demonstrate this property as well.

### 3.2. Network Architecture and Parameters

The input and output layers for this simulation had 8 units: four for the A- and B-items, 4 for the x-items. A depiction of the network is shown in Figures 2 and 3.

**Table 1. Design for Localist Simulation 1**

<table>
<thead>
<tr>
<th>Training Corpus</th>
<th>Consistent, Trained x-items</th>
<th>Inconsistent, Trained x-items</th>
<th>Consistent, Novel x-items</th>
<th>Inconsistent, Novel x-items</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 x1 B1</td>
<td>A1 x1 B1</td>
<td>A1 x4 B1</td>
<td>A1 x4 B2</td>
<td></td>
</tr>
<tr>
<td>A2 x1 B2</td>
<td>A2 x1 B1</td>
<td>A2 x4 B1</td>
<td>A2 x4 B2</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.2.1 Number of Hidden Units

Two aspects of the network were manipulated in this first simulation, in order to better understand how the network performs the nonadjacent dependency-learning task, and to investigate how architectural differences and parameter settings affect this learning. The first aspect to be manipulated was the number of hidden units the network had. Manipulating the number of hidden units typically has two kinds of effects (or one effect, but two different ways of talking about it).

First, it affects the ability of the network to find any solution to the task. The hidden layer is the only place where the network has the ability to combine bottom-up information about current input with top-down information about the previous context (via the hidden layer’s incoming
recurrent connections). For the network to be able to learn to represent nonadjacent structure, it must have enough hidden units to logically be able to solve the problem.

**Figure 2.** A depiction of the network architecture used in Simulation 1, with 3 hidden units.

**Figure 3.** A depiction of the network architecture used in Simulation 1, with 10 hidden units.
For example, in the problem set described above, the SRN must have at least three hidden units or else it cannot learn the nonadjacent dependencies. With less than three hidden units, the network cannot come up with a set of weights and activations allowing it to know to (1) probability match the X’s after an A, (2) activate a B after an X, and (3) know which B to activate based on the recurrent connections feeding back the hidden layer’s activation from the previous time step, telling it which B was on. Effectively, the network needs one hidden unit telling it to equally activate the X’s, one hidden unit telling it when to activate B1, and one hidden unit telling it when to activate B2. As a consequence, an SRN with less than three hidden units would be unable to optimally represent the training set.

The second way the number of hidden units affects network performance is in terms of the qualitative nature in which the network solves the problem. As noted above, with three hidden units the network can (and must) solve the problem in a very discrete way, with each hidden unit effectively coding for one of the logical relations involved in solving the problem. But with more hidden units the network will not need to solve the problem in this fashion. A network with a large number of hidden units (10 for example), rather than activating each hidden unit in a discrete, binary, way, can solve the problem by using a combination of hidden units that are activated to a significantly lesser extent. In the three hidden unit (HU) network in Figure 2, the correct B unit is activated by input from a single hidden unit. Figure 3 shows a network that achieves this same result (activation of the correct B node) via the joint activation of 10 hidden units that would only need to be activated to 10% of the level of the one hidden unit in Figure 2.

The ability of a network with a larger number of hidden units to represent knowledge in a distributed fashion has several consequences. First, networks with more hidden units are more likely to actually solve the learning problem. More hidden units means that more of those units stay, for longer periods of time across learning, in a state closer to their original “neutral” state (because they didn’t need to be modified in order to get all of the output units to activate in the correct way). This tends to mean that networks with more hidden units are less likely to get trapped in local minima while learning, and more importantly, that they are less likely to stay trapped, because if one subset of hidden units has been trained to (poorly) solve the problem, there are other hidden units which the network can use to get out of its suboptimal state. This can have important consequences for some networks, such as those in Simulation 2, where the distance between the nonadjacent items is manipulated. Having more hidden units stay neutral for a longer period means the network will have a bigger chance of having the activation of a string’s initial input (its A-item) still leaving a trace in the hidden layer after a larger number of successive elements.

In addition to better learning, a network with more hidden units will be better able to transfer knowledge learned about one set of items to a new set of items. A network that is representing knowledge as a large set of small weights will be more impervious to that knowledge being disrupted when aspects of the input change. Thus, a network that has learned that A1 predicts B1 using a small number of hidden units may be relying strongly on weights coming to and from the various x-nodes in order to make that prediction, in contrast to a network with a larger number of hidden units. Thus, a hidden layer with more units should have an easier time learning more distant regularities, should tend to do so in a more distributed fashion, and should be more likely to show transfer to items with novel intervening items. In this first simulation, in order to understand the effect of different sizes of hidden units, the size of the hidden layer was varied using the values 3, 4, 7, 10, and 25.
3.2.2 Learning Rate. The second aspect of the network that was manipulated was its learning rate. The learning rate is a measure of how strongly each weight is adjusted when that weight is found to be contributing to error in the network’s predictions. For example, perhaps on a given item, a weight’s “optimal” setting (in order to have produced the correct output) would have been 0.5, but the network’s actual weight setting was 0.2. A learning rate of 1.0 would change the weight by that entire difference, changing the weight by 0.3, from 0.2 to 0.5. If the network’s learning rate were instead 0.5, it would change the weight in the correct direction (making it stronger), but only by half that amount, from 0.2 to 0.35. In many connectionist networks, learning rates are set very low (0.1, 0.01, 0.001, and sometimes lower). The point of lower learning rates is to diminish the effect that any single trial has on the network’s performance, so that the network does not get too fixated on the consequences of a single item. If some of the network’s inputs have contradictory relations, networks with high learning rates will just cycle back and forth, always over-adjusting to the current input without finding a solution that jointly solves both problems. This would be especially likely to happen in the current situation, where any given x-item will be equally likely to be followed by all B-items. The network needs to eventually learn to “ignore” the x-item (or to use it to know it needs to activate a B-item, but not to use it in order to guess which B-item). If the network were strongly reacting to which B followed each x, it would quickly over commit to specific x-B associations and have a harder time learning to ignore them. In this first simulation, the learning rates 0.01, 0.1 and 0.5 were parametrically manipulated to measure their effect on the network’s performance.

3.3. Assessment

The model (and its performance using different parametric settings) was assessed in two ways. First, did the model reach criterion learning (e.g. perfect performance in terms of the optimal posterior probabilities for all items in the training corpus) and how long did it take to get there? If not, how close did it get? Second, how did the model perform on the test items involving novel intervening x-items?

3.4. Hypotheses

Using the training distribution described above, human learners (both infants and adults) have trouble learning the nonadjacent structure present in the training corpus. Thus, if a recurrent network is modeling human performance, learning this nonadjacent structure should not be a trivially easy task. However, there is no human data showing whether or not immense exposure to this exact training corpus would lead to learning the nonadjacencies. The closest analogue is Cleeremans and McClelland’s study in which participants came back for six sessions involving thousands of trials, and did see slow but gradual improvement in learning a grammar that contained nonadjacent relations. Therefore, the closest Simulation 1 can come to speaking to human data is to be able to learn the grammar, but for it to have a nontrivial amount of difficulty doing so. More importantly, the network’s performance in this simple design, but relatively difficult learning situation, will serve as a benchmark for the rest of the simulations. Additionally, only models that demonstrate transfer to test items can be said to qualitatively match human performance on this task.

3.5. Results and Discussion
The results from Simulation 1 are shown in Figures 4-8. Figure 4 shows the network’s activation of the correct B-unit on trained items (e.g. the activation of B1 for the sequence A1-x1-B1), for networks trained with a learning rate of 0.01. The different colored lines represent models with different numbers of hidden units. Figure 5 shows the same model’s performance on novel (untrained) items (e.g. the activation of B1 for the sequence A1-x4-B1, for a model that never saw a sequence containing x4 during training).

Figures 6 and 7 show the trained and novel performance using a learning rate of 0.1, and Figure 8A and 8B show the trained and transfer performance for a model using a learning rate of 0.5. In all cases, optimal performance is complete activation (1.0) of the correct B unit. An activation of 0.5 (especially activation that persists at that level over time) is indicative of a model that was activating both B nodes (the correct one and the incorrect one) to equal levels of 0.5. These were models that had learned that one of the B nodes was supposed to occur in the final position based on the adjacent transition probabilities, but had not learned which B was correct based on the nonadjacent relation. A final note is a reminder that all lines on all plots are the average performance on 30 different runs of the model, starting from different random weight initializations. In all cases, the standard error (variance) of the models were so small that plotting error bars was not helpful (or visible).

As shown in Figure 4 and 5, networks with low learning rates showed very stable performance, with their speed at learning to activate B units, and their speed learning to activate the correct B unit, increasing as they had more hidden units. Networks with 25 hidden units took approximate 450 trials to activate both B units to 0.50, and took approximately another 100 trials to begin reliably distinguishing the correct B unit from the incorrect B unit. All the models with fewer hidden units took approximately 800 trials to reach 0.50 activation for both B units, with big differences in how long it took the models to begin predicting the correct B. Models with 10 hidden units took about 900 trials, models with 7 hidden hits took about 1800 trials, and models with 3 and 4 hidden units were only barely (and unreliably) more likely to predict the correct B unit after 3000 trials. Limited exploratory runs found that both of these models eventually predicted the correct B unit, after about 5000 and 8000 trials, respectively.

Why does the number of hidden units affect how quickly the model learns the nonadjacency? This is a clear case where connectionist models differ from the account offered by traditional models of memory. More hidden units are not equal to more memory in a traditional sense. In this situation, having more hidden units is actually helping the models learn more quickly because it allows more weights to be adjusted at once. Consider a model with only three hidden units. It has three weights (one from each unit) going into each B output unit. The model needs to get the correct B’s output up to 1.0, and must use only these three sources of input to do it. If activation coming into the correct B from each of these hidden units starts off low, then because of the low learning rate, it takes many iterations through the training set to change these three weights enough to get them all high enough to lead to high activation in the B unit. The problem is exacerbated by the fact that the network probably cannot use all three weights to increase both Bs’ activations. In the minimal case (3 hidden units), at least one of the hidden units must be dedicated to only one of the Bs (and a second hidden unit for the other B), so that the network can differentially activate each B in the correct circumstance, but not always activate both Bs. The third hidden unit (in addition to being the unit that the network uses to activate the X output units) can be used to “amplify” the signal coming into the B units so that they more quickly reach threshold activation. Thus, with effectively only two weights going into each B unit that
can be adjusted on each trial, the network takes a very long time to get the correct B units to activate. With more hidden units, the network has more weights that it can adjust on each trial. The network with 25 hidden units could, in principle, use only two hidden units to differentiate the correct B’s, and use all 23 other hidden units as “amplifier” units to quickly raise the activation of both B’s to 0.50. Because, the network will get to this point quickly, using a combination of many different hidden units, most of these hidden units will still be relatively neutral (close to 0.5) compared to the network with three hidden units. As such, once the network has gotten both B units to 0.5, it is more easily able to repurpose some of these hidden units as units that distinguish the two B’s from each other.

**Figure 4.** Networks’ activation of the correct B unit on training items, as a function of the trial number and the number of hidden units, when trained with a learning rate of 0.01.

**Figure 5.** Networks’ activation of the correct B unit on transfer (novel) items, as a function of the trial number and the number of hidden units, when trained with a learning rate of 0.01.
As the learning rate goes up to 0.1 (Figure 6 and 7), network behavior changes considerably. All of the networks reach 0.5 level of activation of both B’s much more quickly. Networks with a large number of hidden units (7, 10, and 25) initially break out of the probability matching behavior in order pick the correct B unit, before crashing back down to chance performance (and they never again go back up, even after 10,000 trials). Models with fewer hidden units (3, 4) never get more than trivially better than learning to pick one of the B’s, but not the correct B. This makes sense given the effect of hidden units that was described above. Models with more hidden units will tend to have more of the hidden units relatively neutral when they get to the B = 0.50 state, making it easier for them to repurpose the hidden units from activating both B’s to activating only the correct B. However, with higher learning rates, they have fewer units in less neutral states than the networks with a lower learning rate. As a result, finding some combination of hidden units that will activate each of the B’s correctly without activating the wrong B is difficult (and even more difficult the smaller the number of hidden units). The networks’ ability to get beyond 0.5 for the correct B drops as a function of the number of hidden units that it had. However, even the network with the most hidden units is unable to fully succeed at the task over the long run. Having used a high learning rate to adjust too many of the hidden units too much too fast, the network is never able to find a way to combine them in away to perfectly attempt to predict only the correct B. Over time, as the network tries more and more to do so, the hidden units become more and more polarized, decreasing the chance of the network ever finding a solution. As the hidden units become more polarized, the noise of the network’s predictions also markedly increases, as once sees happening after about trial 1000.

Figure 6. Networks’ activation of the correct B unit on training items as a function of the trial number and the number of hidden units, when trained with a learning rate of 0.10.
Figure 7. Networks’ activation of the correct B unit on transfer (novel) items as a function of the trial number and the number of hidden units, when trained with a learning rate of 0.10.

At even higher learning rates (0.5 in Figure 8, left) this trend is taken to its natural completion. In these networks, no (reasonable) number of hidden units are able to save the network from overcommitting the hidden units. All of these networks very quickly achieve the ability to very quickly predict that one of the B units should be next, but none of them ever learn the ability to predict which one.

The final point of discussion is how the networks of various learning rates and hidden units perform on transfer items. In brief, networks with very low (0.01) and very high (0.5) learning rates both perform identically on the trained items and on the transfer items. The low learning rate models perform equally well on both types of trials, whereas the high learning rate models perform equally poorly. Only the models with a moderate learning rate perform notably differently, with their transfer performance crashing down to chance more quickly than their test items crash to chance.

Which model (or set of model parameters) best captures human performance, is a question that is rather difficult to answer. The learning trajectory of people in this type of learning task is not something that is regularly published. Instead, all we have are point estimates (data about how people are performing at specific points in time). We know that at short amounts of training (60-120 trials) that people do not show evidence of learning the nonadjacent dependency in a 2x3 (2 AB pairs with 3 intervening x-items) AxB grammar. And we know that after about 2000 trials using a slightly more complex grammar, participants do learn a nonadjacent dependency. Thus, we would expect human performance on this simple grammar to rise above chance at some point, presumably before 2000 trials. This rules out models such as those with high learning rates or very low numbers of hidden units, as these models either do not learn, or take extremely long to do so. At the present, the human data is not sufficient to say whether or not performance is
more like model with low learning rates (slowly climbing to optimum performance without much noise) or like those with high learning rates (noisy performance with an eventual crash back down to chance performance). However, given that this particular peak-then-crash pattern has not really been observed in people, future models will be run under the assumption that human performance is most like those shown in Figure 5, the models with low learning rates and high numbers of hidden units.

**Figure 8.** Networks’ activation of the correct B unit on training items (left) and transfer items (right) as a function of the trial number and the number of hidden units, when trained with a learning rate of 0.5.

### 4. Simulation 2: The effect of variable distances between A→B dependencies

In the Simulation 1, the distance between the nonadjacently related items was always one item. This is useful for a controlled test and exploration of how SRNs deal with nonadjacent dependencies. But in reality, any model of linguistic knowledge will need to be able to track dependencies of much greater distance. Even more importantly, a realistic model of linguistic knowledge would need to be able to represent nonadjacent dependencies of variable length. The dependency between an article (like the) and its subsequent noun is sometimes adjacent, but sometimes separated by a number of intervening words (usually adjectives). This distance-invariance of knowledge about dependencies in language is clearly a critical aspect of the language representation system.

Distance invariance of certain relations in language is also a prime reason why many linguists and cognitive psychologists have argued that language must be based on a system of generative rules rather associations, and further, that these rules must be innate rather than acquired (Chomsky, 1957, 1959; 1965, 1980; Pinker, 1984, 1998; Pinker & Prince, 1988; Marcus, 1999; Fodor & Pylyshyn, 1989). In one variant of this argument, the “Poverty of the Stimulus” (Chomsky, 1980), children receive overwhelmingly positive evidence that certain linguistic constructions are legal, and rarely receive negative evidence of all the linguistic strings that are not grammatical, and as such, never receive enough information to be able to infer the rule about what the actual underlying relation or rule is. As a consequence, Chomsky argues, knowledge about these rules must be innate. The problem of learning nonadjacent dependencies
is exactly one such example of this problem (Chomsky, 1980; Waxman & Lidz, 2002). Children may receive many individual examples of a particular nonadjacent dependency (A-x1-B, A-x1-x2-B, A-B), at different distances, but these various individual associations are never sufficient to infer that, in general, A implies B. Even worse than never being able to learn the abstract A-B relation, the fact that children’s experience with the A-B pairs comes across a vast range of different distances may make it more difficult to learn that A ever predicts B, much less that it always does.

As a consequence, the ability of an SRN to learn nonadjacent dependencies of variable distances is a critical question. Will “noise” (e.g. variation) in the distance between an A and B that are otherwise perfectly predictive of each other, disrupt the ability of the SRN to learn the relation at all? And can an SRN that is trained on nonadjacent dependencies of one length learn to transfer that knowledge to strings of different lengths?

There is also a question about how people actually learn dependencies spanning different distances, and what learning trajectories in these situations imply about the underlying mechanism. Experimental data on this issue is sparse, but there are a few interesting studies worth mentioning. Cleeremans and McClelland (1991) presented participants with a sequence-learning task where the sequence was following an artificial grammar that contained nonadjacent dependencies. They found that participants could learn this nonadjacency, but that performance on the nonadjacent structure dropped off sharply as a function of the distance between the nonadjacently related items, a fact which they successfully simulated using a simple recurrent network. Misyak and Christiansen (2008) replicated this general finding using a simpler grammar. They gave adults an A-x-y-z-B grammar (three intervening items), and found that after training, participants were above chance at judging the grammaticality of strings that either followed or violated the A→B contingency. Additional evidence comes from experiments testing adult performance on natural language involving nonadjacent dependencies. These studies have found that the distance between nonadjacently related items has an effect on processing difficulty. For example, Christiansen and MacDonald (2009) found that people have trouble comprehending recursive sentences, and that this difficulty increases as the distance between the nonadjacently related elements increases. Therefore, it is reasonable to suggest that for the recurrent network to be said to be modeling human performance, it ought to show effects of increased difficulty learning these nonadjacencies as the distance between them increases.

Finally, there is some limited behavioral evidence about how variation in distance between nonadjacently related items may matter over the course of learning. Lany and Gómez (2008; Lany, Gomez, & Gerken, 2009), in a series of studies with adults and infants, showed that prior knowledge of word pairs occurring in adjacent dependent relations helped participants learn nonadjacent relations involving those pairs. This situation occurs frequently in natural language, such as when the adjacent relationship between articles and nouns and the concordant situation where the article and noun are separated by an adjective. This effect could be construed as evidence that knowledge of nonadjacent structure is relatively distance invariant. In addition, however, this evidence can also be used to argue that differences in the distance between dependent items are not purely abstract and distance-invariant. If prior knowledge of an adjacent dependency is a necessary precursor (or at least a huge help) for learning a nonadjacent dependency, this suggests that something about the distances between items affects how this knowledge is learned and represented.

It is therefore an important question whether an SRN, trained on nonadjacent dependencies of one distance, will be able to transfer that knowledge to other distances. For an SRN to be said
to acquire knowledge of linguistic structure, and for it to characterize human performance in learning and representing this knowledge, it must demonstrate two features. First, it must be able to learn nonadjacent dependencies of different lengths, while showing progressively more difficulty as the distance between the nonadjacent dependency increases. Second, it must be able to generalize knowledge that it learns about a nonadjacent dependency to other distances, and in line with Lany & Gómez, show facilitation from getting strings of different lengths rather than having that variation act as noise that disrupts learning.

4.1. Training Corpus, Test Items, and Design

The design for this simulation parametrically manipulated the distance between the A-B pairs. The training corpus had four nonadjacent A-B pairs, and a set of 6 possible intervening x-items. The distance between the A-B pairs was manipulated by varying the number of x-items coming between the A-B pair, ranging from a distance of zero (e.g. A1-B1) to three (e.g. A1-x1-x3-x5-B1). These conditions will be referred to as Span0, Span1, Span2, and Span3. In addition to these four models that were trained on strings that were always the same length, one model was trained on strings of all distances (0-4), a condition that will be referred to as SpanX. This SpanX training condition was used to test the question of whether receiving variation in the distance between nonadjacent dependencies would disrupt or assist learning. The full set of training items for all conditions is shown in Table 2.

For each model there were two different x-items that could occur in each intervening slot, and each intervening item always appeared in the same slot. Thus, models in the Span1 condition always contained the intervening items (and only the intervening items) x1 and x2, occurring in between the A- and B-items. Models in the Span2 condition always had the intervening items x1 and x2 occurring in the first position after each A-item, and always had the items x3 and x4 occurring in the second intervening slot. Models in the Span3 condition added the intervening items x5 and x6 as possible items in the third intervening slot. In all Span conditions, the transition probability for each adjacent dependency was always 0.50 (e.g. in Span1, A1-x1 and A1-x2 were equally frequent, and x1-B1 and x1-B2 were equally frequent), with the exception of the Span0 condition, where the adjacent transition probabilities were 1.0 (A1 always preceded B1, and A2 always preceded B2).

4.2. Network Architecture and Parameters

The input and output layers for this simulation had 10 units: 4 for the A- and B-items and 6 for the x-items. Based on the findings in Simulation 1, the network architecture had 25 hidden units, and the network was trained with a learning rate of 0.01. As in all the simulations, the network was trained with momentum (set to 0.8) and with a gain setting (the slope of the sigmoid function used to calculate net activation) of 1.

4.3. Assessment

As in Simulation 1, 30 runs of each training condition were run that started with different weight initializations, with results averaged across all 30 runs. The networks were then tested in the same manner as Simulation 1, looking at the activation of the correct B output unit, given the x unit immediately preceding the B unit. At each point in training, the network was also tested on
sequences from the other Span conditions, to test the extent to which learning about the nonadjacent dependency in the model’s training condition was transferring to models with different spans.

4.4. Hypotheses

If the models are representative of human learning of nonadjacent dependencies, and more broadly, representative of human knowledge of linguistic structure, two qualitative patterns should be observed. First, the network should have increased difficulty learning the dependency as a function of the distance between the related items. Second, the model should show some ability to generalize nonadjacent dependency knowledge from one distance to another distance.

### Table 2. Test Strings for Simulation 2

<table>
<thead>
<tr>
<th>Test Strings</th>
<th>Span0</th>
<th>Span1</th>
<th>Span2</th>
<th>Span3</th>
<th>SpanX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 B1</td>
<td>A1 x1 B1</td>
<td>A1 x1 x3 B1</td>
<td>A1 x1 x3 x5 B1</td>
<td>A1 B1</td>
<td></td>
</tr>
<tr>
<td>A2 B2</td>
<td>A1 x2 B1</td>
<td>A1 x1 x4 B1</td>
<td>A1 x1 x3 x5 B1</td>
<td>A2 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x1 B2</td>
<td>A1 x2 x3 B1</td>
<td>A1 x1 x4 x5 B1</td>
<td>A1 x1 B1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x2 B2</td>
<td>A1 x2 x4 B1</td>
<td>A1 x1 x4 x5 B1</td>
<td>A1 x2 B1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x1 B2</td>
<td>A1 x2 x3 B1</td>
<td>A1 x2 x3 x5 B1</td>
<td>A2 x1 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x2 B2</td>
<td>A1 x2 x4 B1</td>
<td>A1 x2 x3 x6 B1</td>
<td>A2 x2 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x1 B2</td>
<td>A2 x1 x3 B2</td>
<td>A2 x1 x3 x5 B1</td>
<td>A1 x2 x3 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x2 B2</td>
<td>A2 x1 x4 B2</td>
<td>A2 x1 x3 x6 B1</td>
<td>A2 x2 x3 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x1 B2</td>
<td>A2 x2 x3 B2</td>
<td>A2 x1 x4 B2</td>
<td>A2 x2 x4 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A2 x2 B2</td>
<td>A2 x2 x4 B2</td>
<td>A2 x1 x4 x5 B1</td>
<td>A2 x2 x4 x5 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A2 x2 x4 x6 B2</td>
<td>A2 x2 x4 x5 B2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A2 x2 x4 x6 B2</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Test Strings for Simulation 2
4.5. Results and Discussion

The results from Simulation 2 are shown in Figure 9, 10, and 11. Figure 9 shows the activation level of the correct B unit for models trained in Span0, Span1, Span2, and Span3 training conditions. Each plot shows a model trained in one of the Span conditions, and how well that model transferred performance to items from the other four Span training sets. The top left figure shows the average activation of the correct B unit for models trained on Span0 items (adjacent dependencies). Models trained on Span0 learned the adjacent dependency very quickly, but did not show any transfer to items from different Spans. The other three figures show the activation levels for models trained on Span1, Span2, and Span3. As shown in the figures, all of these models eventually learn the nonadjacent dependency for the Span on which they were trained, taking progressively longer to do so as distance between the nonadjacent items increased (the four are plotted on the same graph in Figure 10). Models trained on Span3 were only barely above chance (approximately 0.51) after 5000 trials of training. In summary, for models trained on a single distance, it is clear that SRNs strongly show the human effect of having more difficulty with a nonadjacent dependency as a function the distance grows, but also appear to be unlike in humans in failing to transfer that knowledge to occurrences of the same nonadjacent dependency with a different distance.

Figure 9. Average activation of correct B unit for models trained on Span0 training set (top left), Span1 training set (top right), Span2 training set (bottom left) and Span3 training set (bottom right). In all cases, the activation of the correct B item is shown, when tested on items from all Span sets.
However, the model’s performance in the SpanX condition (shown in Figure 10) suggests something much more interesting is going on. The SpanX condition, which contains a mixture of items from all the Spans, initially performs much worse, taking much longer to figure out when to predict B’s (which makes sense given that it is getting a much noisier input about whether a B or another x is likely to follow a particular A or x. The SpanX model takes about 3000 trials to learn to predict B’s consistently at the correct time. However, at this point the SpanX model, in addition to learning to predict a B, has also learned to predict the correct B, regardless of the Span of the string.

It quickly shoots past the 0.5 “chance” level, and reaches criterion (perfect B performance, signaling learning all the nonadjacent dependencies) by about 6000 trials. Critically, the SpanX model learns the longer nonadjacent dependencies (Span2 and Span3) much faster than the models that were trained exclusively on the Span2 and Span3 items. This show that the variation in the SpanX model’s input, while leading to slower learning initially, eventually contributes to a much stronger and more robust learning of the nonadjacent dependencies. It shows that an SRN that is given a variable input (and it should be noted, this would be more representative of the actual structure of natural language) is more easily able to learn that A predicts B, and that this is true across variable distances. The fact that the SpanX network learned longer distance nonadjacencies more easily speaks to the power of having variable input, and the importance of this variability in learning a distance-invariant representation.

The SRNs performance on longer sequences in the SpanX condition raises the question of whether the SRN can transfer nonadjacency knowledge to distances on which it has not been trained, if it has been trained on a variety of distances. Put another way, were the Span0, Span1, and Span2 sequences helping learn Span3, or were they leading to an abstract representation of A-B, such that Span3 automatically worked? To test this question, a final simulation was run. This model, referred to as Span012, was trained only on sequences of Span0, Span1, and Span2, and then was tested on sequences of Span3. This simulation is then a true test of whether the model is learning an abstract nonadjacent dependency that can then be transferred to novel strings of novel lengths that the model has not yet seen. Figure 11 shows the performance of this model on sequences of Span3, along with the performance of models trained on Span0 (exclusively), Span3 (exclusively), and SpanX (Span0, Span1, Span2, and Span3). As is clear from Figure 11, the Span012 training allowed the model to do almost identically as well as the SpanX model. Thus, the Span012 model, which had never actually seen a Span3 sequence, was nonetheless able to learn an abstract A-B relationship that it could apply to novel sequences.

How is the SRN succeeding at forming this abstract A-B representation? As described in Simulation 1, in order for the SRN to succeed at all in learning a nonadjacent relation, it must make sure that at least some weights from the A input nodes to the hidden layer stay active. It must then make sure that whatever hidden units were activated by A inputs have strong connections to the B output units.

But this alone is not enough; if it were, the SRN would just always activate the related B immediately after the A, not at the correct point in the sequence. In order for the network to learn both the A-B relation and that it might occur at some distance, it must learn two additional things. First, it must learn that the weights coming in the A-influenced hidden units to the B output units are not the sole thing that should determine if a B turns on. In order to turn a B-output on, the B output units must also have some supplementary input from the x-units that precede them.
Figure 10. Performance of all Span models on their training sets, including the SpanX model.

Figure 11. Performance on Span3 test items for Models that were tested on (1) Span0 items, (2) Span3 items, (3) A mixture of Span0, Span1, and Span2 items, (4) a mixture of Span0, Span1, Span2, and Span3 items.
Finally, the network must learn that when hidden units are activated by A input units, this activation must stay present until the appropriate time to output that activation to the B units. For example, in the sequence A1-x1-x3-B1, the hidden units that are activated by A1 must, via their own recurrent connections back to the hidden layer, maintain some form of activation in the hidden layer that is not obliterated by the input from the x1 input. And it must again maintain this activation when it gets x3 as an input. The network must, as Elman (1990) put it, “learn to remember.”

In this way, the SRN and the way it processes and learns linguistic sequences is fundamentally different than the typical computer metaphor that many cognitive psychologists use for thinking about the mind. In an SRN, there is not set of memory registers or buffers that are pre-determined as places where information will be stored. Instead, that behavior emerges automatically from a network that is doing nothing more than modifying weights such that some combination of weights leads to the correct output. The units that store information in an SRN were not predestined to store information; they could easily have come to be used for some other computation. This point is especially made clear when you consider that the fact that the SRN is “remembering” the A information across time steps is doing so, not by turning a particular hidden unit on and learning to keep it on. Instead it is achieving this by learning a distributed pattern across all 25 hidden units, and that sequences of these distributed patterns eventually lead to the prediction of the correct outcome. Thus, it might be said that while the SRN is learning to remember, it is never learning to have memory (slots). What appears to the everyday observer to be a stored memory is instead the emergent outcome of a complex dynamical system.

The results of Simulation 2 have very significant implications for theories of language learning and representation. The existence of nonadjacent dependencies, and the alleged difficulty of learning and representing them based on association-based mechanisms, is one of the principle arguments in favor of a syntactically structured representation of language. The fact that the occurrence of nonadjacent dependencies in language is very noisy, with the same dependency occurring at a vast range of different distances and in different contexts, is often used as a primary example of the “Poverty of the Stimulus”, evidence that children could not possibly infer the abstract structure of language based on only positive examples from the language. In Simulation 2, we have shown that, rather than this noise proving disruptive for learning a nonadjacent dependency, this noise is actually an essential ingredient in the process. At least, this is true for SRNs and their ability to learn a representation that is abstract and invariant as to the distance between the dependent items. The outcome demonstrated in this simulation dramatically raises the viability of the SRN as a potential model of human language acquisition and representation.

5. Simulation 3: The effect of variability in the set size of intervening x-items

Gómez (2002) discovered another way in which learning was improved by making the AxB artificial grammar less artificial and more ecologically valid. Gómez found that when you manipulate the set size of the intervening x-items, this has a consequence on learning. In the original studies, the finding was that increasing the set size made learning much easier. For example, in a study with infants, infants could not learn the nonadjacencies in an AxB grammar with a 2x3 design (e.g. 2 A-B pairs and 3 intervening items), but that they could learn them in a 2x12 design (2 A-B pairs with 12 different intervening items, see also Gómez and Maye, 2005). Similarly, adults could not learn the nonadjacencies in a 3x2 design, but that they got
progressively better at learning the nonadjacencies when the set size of x went from 6 to 12 to 24. These findings match up well with the thesis that making the situation more ecologically valid should increase learning, because there are a number of cases in natural language that match this situation. In many cases, the nonadjacent dependencies that need to be learned involve a small set of items that are separated by a large set of items. Prime examples are learning the relations between function words and inflections (e.g. is kick ing, but not had kick ing, and other forms of agreement where the number of things that can separate nonadjacently related items is very large).

Gómez’s findings have been extended to show (at least with adults) that the variability effect is u-shaped. Onnis et al. (2012) tested adults on AxB grammars of size 6x1, 6x2, 6x6, 6x12, and 6x24, showing that performance was highest in the 6x1 condition, plummeted to near chance in the 6x2 condition, and then slowly increased as the size of x went up (but with performance in the x=24 condition still significantly lower than in the x=1 condition). Gómez and colleagues have proposed two different (not mutually exclusive) explanations for this phenomenon. The first is that differences in the amount of variability between the various slots in a string cause attention to shift, such that those slots with relatively different amounts of variability are treated differently. Under this story, the middle slot’s difference in variability (both when it is very low and very high) lead to the statistics of that slot being attended to in a different fashion than the statistics of the first and third slot, leading to a higher likelihood of noting associations between slot one and three. The second explanation is that in cases where x=1, learning is trivially easy. In cases where x is very high, learners effectively give up on trying to learn the adjacent dependencies because there is too much to learn, but they do not give up on learning the nonadjacent dependencies, and as a result do ok in the “x is high” conditions as well. In contrast, in the conditions where x is not too high and not too low, they are still trying to track the adjacent dependencies, but cannot succeed due to the higher number of items. And because they never succeed at learning the adjacent dependencies, they cannot move on to learning the nonadjacent dependencies.

The purpose of Simulation 3 is to see whether SRNs are capable of demonstrating the effect shown by Gómez and colleagues. If they do so, then by understanding the ways in which the SRN learns nonadjacencies and shows the set size effect, we may better understand the mechanism that humans are using when faced with this situation in natural language.

5.1. Training Corpus, Test Items, and Design

Simulation 3 used an AxB grammar with four AB pairs. The design for this simulation parametrically manipulated the number of intervening x-items (1, 2, 6, 12, 24). Thus, in the smallest case, the corpus had 4x1 = 4 total strings, and in the maximal case the corpus had 4x24 = 96 total strings. The network experienced all sequences in a blocked fashion, randomized within each block, and was trained for 4800 total trials in all conditions. A consequence of this design (and in the human experiments that used this same design) is that while the total number trials was held constant across all conditions (thus insuring that the model or people had the same amount of experience with each A-B pair regardless of condition), they had different amounts of experience with each AxB string. In the x=1 case, there were only four total strings, and so the network saw each AxB string 4800/4 = 1200 times. In the x=24 case, the network saw each AxB string 4800/96 = 50 times. Across all conditions, the frequency of each A-B occurrence was
controlled, but across the conditions the frequency and variability of each AxB triplet traded off, with frequency higher in the lower x conditions, and variability higher in the higher x conditions.

5.2. Network Architecture and Parameters

The input and output layers for this simulation had 32 units: 8 for the A- and B-items, 24 for the x-items. As in the previous simulations, the networks were trained with 25 hidden units, with a learning rate of 0.01, and with momentum of 0.8.

5.3. Assessment

For Simulation 3, the critical comparison was whether models in different set size conditions experienced different rates of learning the nonadjacent dependency, and whether that difference was qualitatively similar to the human data. The analysis of learning difficulty was assessed by comparing the models’ activation level of the correct B unit at the completion of training (e.g. after 576 trials). The models’ relative activation levels were then compared to the human performance from Onnis et al (2012).

5.4 Hypotheses

If the models in Simulation 3 fit the qualitative pattern of human performance, they should show easiest performance on models trained in the x=1 condition, then with greatest difficulty at the x=2 and x=6 conditions, with steadily increasing performance as x increased (but never again achieving quite the ease of learning as was found in the x=1 condition).

5.5 Results and Discussion

For Simulation 3, 30 models with different randomly initialized weights were run at each level of x (1,2,6,12,24). Each model was run for 576 trials. The mean performance of the models in each condition after 576 trials is shown in Figure 12. As seen in the Figure, in this simulation the SRN did not match the qualitative pattern of the human data. The SRN had the easiest time learning the nonadjacent dependency in the x=1 condition, and had a progressively worse time learning the nonadjacent dependency as x increased. Thus, in the simulation, using the same general architecture and training parameters as in the other simulations, this SRN appears to be a bad fit to the human data.

Why did the SRN fail to behave like human participants? In order to understand how the network was performing and why it behaved the way it did, we performed two follow-up analyses. First, each individual model was compared separately to see if there were any “individual differences” in the models that could be used to understand its behavior. In this analysis, a striking trend emerged. Unlike in the previous models (were no “individual differences” were observed between different runs of the model), the models in Simulation 3 exhibited a strong bimodal tendency. The models either learned easily and quickly, or remained trapped at chance, effectively probability matching the B-items without learning the nonadjacent dependencies. Further, the tendency to exhibit this bimodality was strongly correlated with set size, with the probability of the model failing to learn the nonadjacent dependencies increasing as set size went up. This distribution is shown in Table 3.
Figure 12. The mean performance of 20 models after 576 trials, for level of the set size of intervening items.

Table 3. Counts of the number of runs of each model that succeeded in learning the nonadjacent dependencies, versus those that failed.

<table>
<thead>
<tr>
<th></th>
<th>x=1</th>
<th>x=2</th>
<th>x=6</th>
<th>x=12</th>
<th>x=24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models that Succeeded</td>
<td>20</td>
<td>20</td>
<td>18</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>Models that Failed</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

Why were some models failing? Analyses of the weights in the models that failed showed strikingly similar behavior. All of the models that failed almost immediately set all of the weights from the A input units to the hidden layer to very low levels, which as we saw in Simulation 1, was characteristic of the models’ whose parameter settings led to failure on the simple AxB grammar. And as with those models, these failing models also showed no sign of “learning to remember” the A information. The hidden units whose weights from the A inputs remained nonzero had the same characteristics. They were strongly associated with the x output units (e.g. they were being used to mediate between the A input and the x outputs, or to represent that getting an A input meant that the network should predict an x). These hidden units also tended to have very low weights for their recurrent connections. In other words, the A information was being used to predict Xs, but was not being maintained so as to predict the correct Bs. This contrasts with the models that succeeded, which did so by keeping weights from the A inputs relatively higher, so that A information was more present in the hidden layer. These successful models were the also more likely to keep this A-information active, by having highly weighted recurrent connections from hidden units that tended to be activated by the A-inputs.
Because networks in the “high x” conditions were more likely to fail to learn any nonadjacent dependencies, it is important to consider that these “failing” networks were dragging down the average performance of all of the higher x conditions shown in Figure 12. Figure 13 shows the performance of the networks when only those models that succeeded are considered. As can be seen in this figure, the networks that succeeded showed a qualitative pattern that is very similar to that of the human participants.

![Graph](image)

**Figure 13.** The mean performance of 20 models after 576 trials, for level of the set size of intervening items, broken down by “all models” and “successful models” (e.g. those which were doing better than chance after trial 576).

Why were the models with larger x values more likely to fail to learn the nonadjacent dependencies at all, but also more likely to show the human pattern if they did succeed? Both of these factors actually make sense, in light about what we learned about the performance of SRNs on the AxB grammar in Simulation 1.

With the simple recurrent network, it makes sense that as X increases, the chance of getting stuck in the “probability match the B’s” local minimum would also increase. It basically comes down to how much the network needs to adjust the hidden units in order to probability match when predicting that the Xs will follow the As. If there is only one X to predict (in the x=1) case, the network needs to alter very few of the network’s weights, and as a consequence most of the hidden units are still in a relatively neutral state even after the network has learned about the A-x contingencies. The network is therefore more easily able to find a set of weights that can also concurrently solve the nonadjacency problem. A basic SRN with the learning parameters outlined here will find it much easier to learn adjacent dependencies first. If learning the A-x and x-B adjacent dependencies uses up all the networks resources (e.g. drives too many of the hidden units away from a relatively neutral state), this will make it very difficult for the network to learn the nonadjacent dependencies. The more x’s there are, there more adjustments the network needs.
to make to accommodate all the A-x and x-B relations, and the less likely it is to be able to learn the A-B relations.

But, assuming the network doesn’t get stuck in adjacent dependency local minimum, the fact that the network then tends to learn in manner characteristic of the human u-shape also makes a lot of sense, given the distributional statistics of the input. The x=1 case is almost trivial for the network to learn. It takes the network almost no time to learn to predict that the one x follows every A, and that one of the B’s will follow that X. As a result, this network will have eliminated this source of error almost immediately, with the only source of error left being that which deals with the nonadjacent relationships. And because the network needed to alter very few weights to learn this one x, most of the hidden units are left in a very neutral state, making the nonadjacent dependency relatively easy to learn.

As the number of x’s increases, two factors are changing at once, leading to the u-shaped curve. On the one hand, the number of hidden units that are left in relatively neutral states after learning the adjacent dependencies is dropping; this makes the network have more trouble learning the nonadjacent dependencies, for the reasons already described. But a second factor is working in the other direction, and concerns the ratio of each x to each B in the training set. When x=2, then any particular x (x1 or x2) occurs in half of the items in the data set, whereas any particular B (B1, B2, B3, or B4) only occurs in 25% of the items. Remember that the network is making a weight adjustment on item in every trial: one weight update after A1-x1, another after x1-B1, another after A2-x1, another after x1-B2, and so on. This means that when x=2 and B=4, because the proportion of x’s to B’s is 2:1, the network is making twice as many updates improving the performance of individual x’s as it is updates that improve the performance of individual B’s. This will lead the network to very quickly reach criterion performance on the x’s, with performance on the B’s lagging behind. And if B performance lags too much, the network may have worked itself into a state where the hidden units were relatively less neutral, leading them to take longer to be re-adjusted so as to capture the B information. Thus, in low x conditions, the network takes longer to repurpose hidden units away from being used to predict the A-x and x-B adjacent contingencies to being used to predict nonadjacent A-B contingencies.

In the X=6 case, the ratio is 2:3; now in favor of the B’s, but only slightly. This means that a higher proportion of the weight updates are working on learning to predict the B’s. And as x increases, the ratio becomes more and more in favor of the B’s. When X=12 the ratio is 1:3, and when X=24 the ratio is 6:1. At some point the benefit that B’s get from shifting the X:B ratio outweighs the effect of more x’s taking up more hidden unit resources, and performance on the B’s starts to go up again. At the theoretical limit of the X:B ratio (infinity:4), where the network sees a particular x once and then never sees it again, the network will never be able to make any improvement on the x-items (and all x-based weight adjustments will just cancel out as noise), and therefore all of the network’s learning will be concentrated on learning to correct the appropriate B-item.

In summary, what do these findings say about the SRN’s ability to simulate and explain human performance on nonadjacent dependencies as a function of the variability of the items that intervene between the nonadjacent dependencies? The overall performance shown in Figure 12 was clearly driven by the fact that the probability of failure to learn anything about the nonadjacent dependencies increased as the intervening variability increased. This general fact is not consistent with the human data, and thus is evidence that the exact SRN used in these simulations is not the best model of human performance, especially for this task. But models like
the SRN models should not be ruled out completely, given that they did fit the data very well when they succeeded, and because the reason they showed the human pattern of data when they succeeded made sense, as described above. Given this, it could be argued that the SRN is quite well suited to fit human performance on this task, if a variant of the model is being used that is unlikely to get trapped in the local minimum of immediately (incorrectly) learning not to remember the A information.

A number of investigations of SRN behavior on more complex, natural language input have demonstrated ways in which SRNs can be modified so that they will avoid getting stuck in local dependencies. Allen & Seidenberg (2002) demonstrated on such method (see also Pearlmutter, 1989). In their recurrent networks, rather than treating each input as single discrete event, and only passing information back to the hidden layer once, the network treats the recurrent activation in a more continuous fashion and allows the recurrent feedback to extend “further back in time”. The consequence is that the recurrent feedback at any given point in time is a combination various different previous time points, which allows the network to more easily notice longer distance dependencies. Allen and Seidenberg showed that this type of network dynamic dramatically increased network performance on learning the grammaticality of a miniature English-like artificial language.

Another method was demonstrated by Sibley, Kello, Plaut, & Elman (2008). Sibley et al. created a variant of the SRN called the “Sequence Encoder”, which is effectively two SRNs connected to each other, with a hidden layer that mediates between the two SRNs. The goal of the traditional SRN is to learn to predict each individual item in a sequence. The Sequence Encoder’s goal is different. Rather than trying to predict individual elements, the goal of the Sequence Encoder is to learn to reproduce the entire sequence. The Sequence Encoder takes the entire sequence into the first “input” SRN, and learns a set of weights connecting that SRN to the Sequence Encoder’s internal hidden layer. The goal of this hidden layer is to learn a representation for the sequence as a whole. Then the network learns a set of weights from this hidden layer to the second, “output” SRN, which can be used to output the entire sequence. Because the Sequence Encoder is trying to learn representations for the sequence as a whole, rather than for individual elements, it is much less likely to get stuck learning only the adjacent dependencies. As a consequence, the Sequence Encoder should be much more likely to show the human pattern of performance on this task.

6. Simulation 4: The Effect of Similarity and Consistency Amongst Nonadjacent Relations

Similarity structure amongst constituent items has been found to be a critical cue for learning nonadjacent dependencies, and it is a cue that is present in an overwhelming number of the nonadjacent dependencies a language learner needs to acquire. Studies by Newport & Aslin (2004), and Creel et al., (2005) showed that nonadjacently related items that are perceptually similar to each other (and different from the intervening items) are learned more easily than nonadjacently related items that have no such similarity structure. Likewise, Willits, Lany, & Saffran (2012) found that infants can learn a nonadjacent relation amongst known words if the words are semantically similar (e.g. doggy-x-kitty), but not when they are semantically dissimilar (doggy-x-banana). Given these findings, showing that SRNs are capable of simulating this

1 And in fact, in five pilot runs of the Sequence Encoder at each level of x, no run of the model got stuck learning only the adjacent dependencies, and the sequence encoder’s performance was
similarity bootstrapping would add even more evidence that the SRN could serve as a good model of human language learning.

In addition to similarity overlap amongst nonadjacently related items, another critical property of many nonadjacent dependencies in natural language is that they have “arbitrary but consistent” relationships between semantic structure and syntactic structure. In other words, the words give, child and present are not inherently more similar to each other. But present and the other possible nouns that can take its place as the direct object of give, tend to be similar to each other. This “arbitrary but consistent” mapping between give and its arguments is the basis of this nonadjacent relation. Willits and Saffran (2012a) have shown that adult learners can learn novel nonadjacent dependencies that have arbitrary but consistent correlations between semantic structure and syntactic structure. Can an SRN also show this effect, more easily learning nonadjacent relations when the words in the relation are arbitrary but consistent, compared to a situation where there is no systematicity?

In the following simulation, we tested a SRN’s ability to capture these effects of similarity by using a slightly modified variant of the models in Simulation 1-3. The first three simulations used a purely localist code for each stimulus item; each stimulus was exclusively represented by a single input node and output node. In this simulation, in addition to each stimulus having its own exclusive node, each stimulus also had second, “category” node, which denoted which group or category the item was from. Thus, items’ similarity (perceptual or semantic) was operationally defined as whether or not the two items shared the same category node, in addition to each having their own unique unit.

6.1. Training Corpus, Test Items, and Design

The design for this simulation is identical to that used in Willits & Saffran (2012a), and is shown in Table 4. Note that in Simulation 4, the letters A and B are used to denote items that are from the same category, as opposed to items that appear in the same position in an AxB grammar. Thus, the string A1-x1-A3 would denote a string where the nonadjacently related pair was from the same category, and A1-x1-B3 denotes a pair from different categories.

Simulation 4 had three conditions. In the “Category Match” condition, there were four nonadjacent pairs from two different categories, arranged such that the nonadjacently related pairs were always in the same category. In the “Category Mismatch”, there were again four nonadjacent pairs from two different categories, this time arranged such that the nonadjacently related pairs were always from opposite categories. Finally, in the “No Category Cue” condition, the four pairs were arranged such that half the time the pairs were from the same category, and half the time they were from opposite categories. The result is that in this condition, category membership was not a useful cue to the nonadjacent dependency across the entire set of items.

6.2. Network Architecture and Parameters

The input and output layers for this simulation 15 units: 12 of them designating each of the twelve items’ unique nodes (four A’s, four B’s, and four x’s), and three nodes specifying the three categories (A, B, and x). The network architecture is shown in Figure 14. With one major exception, the rest of the network architecture and parameters were the same as in Simulation 1 and 2. The network had 25 hidden units, was trained with a learning rate of 0.01. This one exception was the activation function that was used to calculate the output nodes’ activation
level. In the previous simulations, the “SOFTMAX” normalization function was used. The SOFTMAX function normalizes the output of the units so that they sum to one, which can effectively be used to interpret the activation of the nodes as the network’s estimation of the posterior probability of that element appearing next in the sequence. In this simulation, because more than one unit is supposed to be turned on for each stimulus (both the category node and the item-specific node), the SOFTMAX function is no longer appropriate. Instead, the activation is just the unnormalized output of the sum of all the input into each node, transformed into a value between 0 and 1 according to the nonlinear sigmoid function in Formula 2.

**Output Layer**

**Hidden Layer**

**Input Layer**

*Figure 14.* The network architecture used in Simulation 4.

6.3. **Assessment and Results**

The model was assessed in two ways. First, for a given item in the third position that the network was trying to predict, the activation level of five nodes was compared: (1) the activation level of the node specifying the category that the target item was from, (2) the activation level of that item’s the unique node, (3) the activation level of the incorrect item from the same category
as the correct item, i.e., if the network was trying to predict A3, what was the activation level of A4, (4) the activation level of the node specifying the wrong category, and (5) the average activation level of the two nodes specifying the wrong item from the wrong category (i.e., if the network was trying to predict A3, what was the average activation of B3 and B4). It should be noted that “chance” performance was not equal in all these cases. For the category nodes, there were only two possible nodes that were likely to be on; as such if a network knows that one or the other should be on, but isn’t sure which one, it will output a value of 0.5 for both category nodes. For the item-specific nodes, there were four possible choices (A3, A4, B3, and B4). As such, if the network has learned the adjacent dependencies but not the nonadjacent dependencies, it will activate each of these nodes equally, to 0.25.

Table 4. Design for Simulation 4

<table>
<thead>
<tr>
<th>Category Match</th>
<th>Category Mismatch</th>
<th>No Category Cue</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 x1 A3</td>
<td>A1 x1 B3</td>
<td>A1 x1 A3</td>
</tr>
<tr>
<td>A2 x1 A4</td>
<td>A2 x1 B4</td>
<td>A2 x1 B4</td>
</tr>
<tr>
<td>B1 x1 B3</td>
<td>B1 x1 A3</td>
<td>B1 x1 B3</td>
</tr>
<tr>
<td>B1 x1 B4</td>
<td>B1 x1 A4</td>
<td>B1 x1 A4</td>
</tr>
<tr>
<td>A1 x2 A3</td>
<td>A1 x2 B3</td>
<td>A1 x2 A3</td>
</tr>
<tr>
<td>A2 x2 A4</td>
<td>A2 x2 B4</td>
<td>A2 x2 B4</td>
</tr>
<tr>
<td>B1 x2 B3</td>
<td>B1 x2 A3</td>
<td>B1 x2 B3</td>
</tr>
<tr>
<td>B1 x2 B4</td>
<td>B1 x2 A4</td>
<td>B1 x2 A4</td>
</tr>
<tr>
<td>A1 x3 A3</td>
<td>A1 x3 B3</td>
<td>A1 x3 A3</td>
</tr>
<tr>
<td>A2 x3 A4</td>
<td>A2 x3 B4</td>
<td>A2 x3 B4</td>
</tr>
<tr>
<td>B1 x3 B3</td>
<td>B1 x3 A3</td>
<td>B1 x3 B3</td>
</tr>
<tr>
<td>B1 x3 B4</td>
<td>B1 x3 A4</td>
<td>B1 x3 A4</td>
</tr>
<tr>
<td>A1 x4 A3</td>
<td>A1 x4 B3</td>
<td>A1 x4 A3</td>
</tr>
<tr>
<td>A2 x4 A4</td>
<td>A2 x4 B4</td>
<td>A2 x4 B4</td>
</tr>
<tr>
<td>B1 x4 B3</td>
<td>B1 x4 A3</td>
<td>B1 x4 B3</td>
</tr>
<tr>
<td>B1 x4 B4</td>
<td>B1 x4 A4</td>
<td>B1 x4 A4</td>
</tr>
</tbody>
</table>

The second way the networks was assessed was to look at the average of the nodes for the “correct” item (i.e. (1) and (2) above), which was compared to the average of the average of the nodes for the “false alarm from the same category” (i.e. (1) and (3) above), which was compared to the average of the “completely wrong” item (i.e. (4) and (5) above).

6.4 Hypotheses

For the SRN to behave as people do, it ought to learn the nonadjacent dependencies in the Category Match and Category Mismatch conditions significantly more easily than the No Category Cue condition.

6.5 Results and Discussion

Figure 15 shows the performance of the network in the Category Match condition, plotting the activation of the individual nodes in the network, as well as the nodes averaged in order to
show performance for the correct item, the wrong item from the correct category, and the wrong item from the wrong category. Figure 16 shows the same data for the Category Mismatch Condition, and Figure 17 shows the data for the No Category Cue condition. Figure 18 shows the difference scores between conditions.

The simplest summary of the results is that the network learned the correct response (the blue line or lines in any of the figures) significantly faster in the Category Match Condition and in the Category Mismatch condition, and significantly slower than in the No Category Cue condition. There was no significant difference in the blue line(s) in Category Match and Category Mismatch conditions. As such, at a very general level, the model is a good fit to human performance.

Another interesting way in which the model’s performance matched behavioral data was its response to the “false alarm” item, the wrong item from the same category as the correct item. In Willits & Saffran, rates of choosing the false alarms item was higher than rates of choosing the completely wrong item (an item from the wrong category) in both the Category Match and Category Mismatch conditions, but with no difference in the No Category Cue condition. The networks exhibited this same pattern of behavior. The activation level of the false alarm items was higher (and equally high) in the Category Match and Category Mismatch conditions. For example, at the end of training in the Category Match condition, the average activation of the “Correct” item (the blue line in Figure 12A) was about 0.80. The average activation of the “False Alarm from the same Category” was about 0.65, whereas the average activation for the “Wrong Category Wrong Item” was about 0.05. These averages are what they are because both “Correct Category” items share the Category node, which was at 1.0 by the end of training, while the correct item from that category’s node was at about 0.6 and the wrong item from that category’s node was at about 0.3 (and trending down quickly).

This network alone is not a full model of the decision people would make. For that, one would need some way to translate these activation values into actual choice probabilities. One simple decision rule is the Luce Choice Rule (Luce, 1959), where the probability of choosing choice A is the value of that item divided by the sum of the value of all the items, then the probability of a person choosing the correct item would be about 53% \((0.8 / (0.8 + 0.6 + 0.05 + 0.05))\), with a chance of 40% of choosing the false alarm item, and a 6% chance of choosing one of the two wrong items from the wrong category.

This performance is a pretty good match to the participants’ behavior in Willits & Saffran (2011) Experiment 3 Category Match condition. In this experiment, participants saw nonsense words in nonadjacently related sequences (just like in Gómez, 2002), but concurrently saw a picture with each nonce word, providing them with category information about the word’s referent. In this experiment, participants chose the correct item 55% of the time, the false alarm item 30% of the time, and the completely wrong items 7% of the time. However, in this behavioral experiment, participants always had a forced choice between two items, which almost certainly inflated the proportion of times participants chose the completely wrong item relative to the false alarm item. Further research (with a behavioral design that is more easily comparable to the models’ performance) needs to be done in order to comment on the quantitative fit of the model to the behavioral data. But from a qualitative perspective, the model fits the human data quite well.
Figure 15. On top, the activation of the various nodes in the Category Match Condition. On bottom, the average of the two nodes for each item, allowing for an easier depiction of the networks overall accuracy.
Figure 16. On top, the activation of the various nodes in the Category Mismatch Condition. On bottom, the average of the two nodes for each item, allowing for an easier depiction of the networks overall accuracy.
Figure 17. On top, the activation of the various nodes in the Category Mismatch Condition. On bottom, the average of the two nodes for each item, allowing for an easier depiction of the networks overall accuracy.
Figure 18. The difference in activation of the nodes as a function of the training condition. At the top is the difference between the Category Match condition and the No Category Cue condition. In the middle is the difference between the Category Mismatch condition and the No Category Cue condition. At the bottom is the difference between the Category Match condition and the Category Mismatch condition.
Analyzing the performance of the individual nodes in more detail (Figure 15, 16, & 17), it can be noted that in a qualitative sense, the networks in all three conditions started out very similar. All networks very quickly learned the adjacent transition probabilities (demonstrated by reaching 0.5 activation for both category nodes and 0.25 for all of the item nodes) by around trial 300. At this point, however, the Category Match and Category Mismatch networks start to distinguish themselves from the No Category Cue network. These two networks start to almost immediately distinguish which category node they should be turning on (which makes sense as this node is predictive in and of itself, and also correlated and therefore likely highly associated with the item level nodes). The models in the No Category Cue condition were much slower to distinguish which category node should be on for a given trial, which makes sense because this item was not consistently used across items in the same way as in the other two conditions. Likewise, you can see that in the No Category Cue condition, the model never treats the False Alarm items differently from the Totally Wrong items.

To summarize, when the SRN is given any sort of “category” or “similarity” information, it is able to use this information to vastly increase the speed of learning a linguistic dependency, if this category or similarity information is correlated with the linguistic structure the network is trying to learn. In this way, the SRN behaves in a way that is very analogous to the way that humans behave in artificial grammar experiments containing nonadjacent dependencies, where there is perceptual or semantic similarity information that can be used as a cue to learning (e.g. Newport & Aslin, 2004; Creel et al., 2005; Willits & Saffran, 2012; Willits et al., 2012).

The result is quite important for two reasons. First, the behavioral studies themselves are important. The studies demonstrate that complex linguistic structures such as nonadjacent dependencies – one of the principle justifications for syntactically structured representations of language – are in fact easily learnable under certain circumstances. Those circumstances are cases where there is an alignable perceptual or semantic cue, and these cases are very likely to cover a vast range of the actual nonadjacent dependencies in language. By using an SRN to simulate this effect, and demonstrating that the SRN behaves in qualitatively similar manner as human learners, we demonstrate that this type of learning can be performed by relatively simple, association-based learning principles.

Second, successfully using the SRN to simulate these behavioral results is useful because it helps clarify potential mechanisms behind this learning. Newport and Aslin argue that the perceptual similarity of the nonadjacent items may lead to a Gestalt-like “pop out” effect, whereby nonadjacent items become more salient, allowing relations among those items to be more easily learned. There has, in fact, been a long debate over these kinds of perceptual pop-out effects, and whether or not they can be explained by association-based principles. Relatedly, stating that an effect is due to a “shift in attention” is arguably not much of an explanation of the phenomenon. It is more of a re-description of the effect. The question is, why and how does attention shift. Using an SRN model allows us to analyze how and why the SRN shows this similarity-based effect, and to propose and further test this as the mechanism. In this case, the reason the SRN is succeeding is somewhat complex (the SRN is, after all, a complex dynamical system). But its behavioral can be interpreted in light of some facts about its performance that are clear in Figure 15.

The first step is to consider the qualitative nature of the networks’ behavior, especially the difference between the two category-consistent conditions and the No Category Cue condition. All three models very quickly initially learn the set of possible B-items, and effectively
probability match between them. All three networks do this for both the item-specific nodes, and for the category nodes. One way to think about this is that the network is tracking two kinds of information at the same time: both the sequential structure in terms of the categories, and separately, the sequential structure in terms of the actual items. To use an example from Willits et al., it’s as if the network is learning both the item sequence (dog-towel-kitty) and also the category sequence (Animal-BathroomObject-Animal), in the Category Match condition. Models in the Category Match and Category Mismatch conditions more quickly start learning the nonadjacent dependency, but what they are learning first is the nonadjacent category sequence. Before the network is getting dog-x-kitty correct, it is getting Animal-x-Animal correct. And then, in the Category Match condition, because the Animal-x-Animal (and Food-x-Food) sequence is consistently and perfectly correlated with the individual items, this association between the item stream of information and the category stream of information helps the model learn which specific item should end the sequence. This contrasts with the No Category Cue condition, which takes longer (about 500-1000 trials) to correctly learn the category sequence, which it is learning perfectly in synch with the item sequence. It cannot use one to bootstrap the other because the two sequences are uncorrelated.

This explanation is somewhat different than that proposed by Newport and Aslin, and doesn’t need to make any attributions to shifts in attention. Instead the proposed mechanism is that there is information available to the learner of two different types, or at two different levels of abstraction, and that the learning system is able to track sequential dependencies in both of these streams concurrently. In cases like the Category Match or Category Mismatch conditions, the fact that the information in the category stream of information is more consistent, regular, and simple than in the item stream (or than in the category stream of the No Category Cue condition), makes it such that this information will be learned more easily. Then, because this more easily learned information was also correlated with the item information in the Category Match and Category Mismatch conditions, this makes it possible for this semantic information to aid in learning the item stream of information.

7. Simulation 5: Learning Abstract Rules

The final simulation deals with an issue that goes straight to the heart (at least in some people’s minds) of whether or not association-based models such as SRNs are capable of representing language and linguistic behavior, as well as other aspects of higher cognition. The issue is whether or not a connectionist network can be used to represent the “rule-like” nature of language and cognition. According to many classical theories in linguistics, language is, by definition, a system comprised of rules (i.e. in English, a sentence is comprised of a noun phrase followed by a verb phrase, or the verbs that go with singular nouns must also be singular). In the early days of cognitive psychology, it was argued that the rule-based nature of language excluded association-based theories of language acquisition and representation, as those theories were incapable, in principle, of learning abstract rules (Chomsky, 1957, 1959; Bever, Fodor, & Garrett, 1957). With the increased popularity of connectionist models in the 1980s and 1990s, these arguments were revisited, with debate involving two key issues. First, whether or not connectionist models were identical to simple associationist theories, and were therefore also in principle inadequate as models of the mind (Fodor & Pylyshyn, 1988; Lachter & Bever, 1988; Pinker & Prince, 1988). Second, a number of researchers have argued that even if connectionist
models are doing something more substantial than simple associationism, they are still fundamentally incapable of behaving in the way that people do.

A number of articles by Marcus (Marcus, 1998a, 1998b; Marcus et al., 1999) have emphasized this argument, using a series of behavioral studies and theoretical analyses of connectionist networks like the SRN, to make a clear argument that there are certain types of structure in language that connectionist models cannot possibly learn. One such argument (of which Marcus provides many variants) is that connectionist models have no possible way to generalize abstract knowledge that they learn from one set of items to a new set of items. For example, in a study with seven-month-old infants, Marcus et al. showed that infants who are exposed to one set of syllables occurring in a repetitive ABB pattern (e.g. strings of three syllables where the second and third syllables are the same) appear to learn, not just associations between items, but the abstract notion that items are repeating. Infants exposed to this ABB pattern during a familiarization phase discriminate between new sequences that continue to follow the ABB pattern and new sequence that follow an ABA pattern. The infants can perform this discrimination even when the new strings are comprised of an entirely different set of sounds. Marcus et al. argued that because there was no overlap in the items in the two phases of the experiment, this rules out association-based explanations of the knowledge, and that the only possible explanation is that the infants learned a rule.

Marcus (1998a, 1998b) argued that connectionist models are inherently incapable of explaining this kind of finding. Marcus’s specific argument is as follows. Imagine the SRN one would use to learn the initial ABB pattern. The model would learn this ABB pattern for the initial items quite easily, by learning a set of weights from the A-items and B-items in the input layer to the hidden layer, and a set of weights from the hidden layer to the A- and B-items in the output layer, such that it made the proper predictions about which item was likely to come next in a sequence. But now imagine this network being tested on the new set of items that follow either an ABB and ABA pattern. The weights coming from and going to the new items have not yet been trained, and so therefore the network cannot possibly use them to generate correct predictions for the new units. To Marcus and many others, this constitutes elemental proof that connectionist networks cannot, by definition, learn an abstract rule that they can transfer from one set of stimuli to another. Marcus’s argument was really just an unacknowledged repackaging of the Terminal Meta Postulate argument (Bever, Fodor, & Garrett, 1961), whereby it was argued that associationist systems couldn’t, by definition, learn something that is abstractable and distinct from the items over which it is learned.

Marcus’s argument generated much controversy, and a number of papers were published demonstrating that connectionist models could learn the repetition (ABB) and alternation (ABA) patterns (Altmann & Dienes, 1999; Christiansen & Curtin, 1999; Elman, 1999; Gasser & Colunga, 1999; Negishi, 1999; Shastri, 1999; Schultz & Bale, 2001). All of these were an extension of a simple point by Seidenberg & Elman (1999), who argued that networks’ inability to represent hidden rules was irrelevant, as there was no need for the infants in Marcus et al.’s study to use abstract rules in the first place. There were a number of features and factors in the input that made it possible for statistical learning principles to explain the behavior, such as exploiting feature overlap in the sounds. It may have been the case that at the item level of analysis, there was nothing in common between training items and the test items, but that rule out the possibility that the infants were making use of correlated microfeatures (such as various phonemic features that training and test items may have shared) in order to transfer knowledge from training to test. These models all demonstrated this fact by using distributed phonemic
feature vectors as inputs and outputs. Thus, even though the same items were not repeated at training and test, various microfeatures were present in both instances that could be used to transfer knowledge about the structure in the training to the test situation.

Marcus and others have criticized each of these models for idiosyncratic factors, such as using unrealistic representations of phonemes, training the model to do a different task than the infants had done (learning to categorize versus learning to predict), and unusual analyses of the network (e.g. Altmann & Dienes’s use of overall output vector similarity rather than correct prediction of the output unit). These criticisms are convincing to some, but not to others. However, all these arguments about implementational factors all seem obscure the fact that Marcus and his critics are mostly talking past each other on this issue. Marcus’s main argument is that connectionist architectures cannot, in principle, learn abstract rules. Marcus’s critics all have replied with some form of a “you don’t need rules” argument. This may be true. But there remain a number of people who are convinced that some form abstract rules remain a critical part of the human mind, and connectionist arguments to the contrary leave them unconvinced. As a consequence, the failure of connectionist models to be able to learn abstract rules leaves these people skeptical about the entire connectionist enterprise.

In Simulation 5, we take Marcus’s criticisms of earlier models of the ABB vs. ABA task at face value. The jury is still out on whether all human language can be explained without recourse to abstract rules, but there do seem to be a number of situations in language and other aspects of higher cognition that are rule-like. Is there any possible way that an SRN could learn something abstract and transferable, “outside the training set”, as Marcus describes it? Can a simple recurrent network using an exclusively localist representation, without microfeatures to exploit, learn an abstract structural relation about its input and transfer that abstract structural knowledge to new inputs?

In the following simulation we show that it can, as long as one makes a simple (and more realistic) assumption about the network’s situation: learning never stops. In the real world, people don’t go through discrete phases, where first they learn stuff, and then the learning processes shut down and they test their knowledge. In the real world, learning is continuous. Even in artificial situations like psychology experiments, participants learn things during test phases. In Simulation 5, we transfer this insight to the SRN’s training situation. Rather than training the network on one set of ABB or ABA sequences, and then stopping the learning process and testing it on a new set of items, we allow the model to continue learning during these new sequences. The hypothesis is that, if the network is learning something abstract about repetition and alternation in the ABB or ABA sequences, even though it will not immediately be able to transfer to this knowledge to novel items on its very first exposure to them, it will nonetheless have an easier time learning the new sequences, if they are consistent with the abstract pattern on which the network had been trained.

7.1. Training Corpus, Test Items, and Design

The design for this simulation is shown in Table 5, and was identical to that from Marcus et al., 1999, Experiment 1. In the initial training condition, there were six A-items and six B-items, which were combinatorially combined to create 36 possible ABA and ABB strings. These strings will be referred to as the strings from condition ABA1 and ABB1. The consequence of this design is that within each particular condition, the only overall structure is that the items either follow the alternation rule (in the ABA1 conditions) or the repetition rule (in the ABB1
Both strings’ initial two elements are identical (A first, then B) with each A equally likely in the first position and each B equally likely to follow each initial A. The third element in each sequence is the only one that varies between conditions, and is predictable in the ABA1 sequence as a nonadjacent alternation back to the first element in the sequence, and predictable in the ABB1 sequence as a repetition of the second element.

For the second part of the experiment, an entirely different set of six A-items and six B-items were used, following the same ABA and ABB grammars. These items will be referred to as conditions ABA2 and ABB2. Because the strings in ABA1 and ABA2 shared absolutely no items from between the two training sets, there was no way to use any local association information that was learned in the first training phase, and apply that to the second training phase (and the same was true for the ABB1 and ABB2 items). The network could only succeed if it learned something about the more general and abstract alternation or repetition structure.

As described above, the network was trained in two phases. First the network was trained for 512 trials (14 repetitions of the entire training set) on either the ABA1 or the ABB1 training set. This number of trials was selected because, in initial pilot simulations, it was found that this was the number of trials necessary to get the ABA-trained and ABB-trained networks to criterion performance (i.e. nearly 1.0 activation of the correct A or B item in the final position). After these 512 initial training trials, each of these networks were then trained on either the ABA2 or the ABB2 items, creating a 2 x 2 design (e.g. first training phase: ABA1 vs. ABB1; second training phase, ABA2 vs. ABB2). The critical dependent measure was the rate of learning in the second training phase, to see whether or not consistency in the type of training across the first and second training phases facilitated learning.

As in the previous simulations, 30 different models in each condition were run starting from different random initial weights, and the results presented below show the average across all 30 models in each condition.

<table>
<thead>
<tr>
<th>Table 5. Design for Simulation 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABA1</td>
</tr>
<tr>
<td>A1 B1 A1</td>
</tr>
<tr>
<td>A1 B2 A1</td>
</tr>
<tr>
<td>A1 B3 A1</td>
</tr>
<tr>
<td>A1 B4 A1</td>
</tr>
<tr>
<td>A1 B5 A1</td>
</tr>
<tr>
<td>A1 B6 A1</td>
</tr>
<tr>
<td>A2 B1 A2</td>
</tr>
<tr>
<td>A2 B3 A2</td>
</tr>
<tr>
<td>A2 B5 A2</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>A6 B1 A6</td>
</tr>
<tr>
<td>A6 B2 A6</td>
</tr>
<tr>
<td>A6 B3 A6</td>
</tr>
<tr>
<td>A6 B4 A6</td>
</tr>
<tr>
<td>A6 B5 A6</td>
</tr>
</tbody>
</table>
3.4.2. Network Architecture and Parameters

The input and output layers for this simulation had 24 units, 12 for the A-items and 12 for the B-items. The rest of the network settings were the same as in previous simulations (25 hidden units, a learning rate of 0.01, and trained with momentum of 0.8).

3.4.3. Assessment of Results

The model was assessed to see whether there was a significant difference between the models in their activation of the correct final unit, as a function of their first training phase and their second training phase, especially with regard to whether the two training phases were consistent or inconsistent.

3.4.4 Hypotheses

If the SRN is able to learn and apply an abstract rule such as repetition or alternation, then the networks ought to learn faster during training phase 2 if they are receiving the same kind of abstract structure as they received in training phase 1. In other words, learning in the ABA2 condition ought to be faster for those networks that had been trained on the ABA1 sequences then those that were trained on the ABB1 sequences, and learning ought to be faster on the ABB2 condition that were trained on the ABB1 sequences than for those that were trained on the ABA1 sequences.

3.5. Results and Discussion

Figure 19 shows the SRNs performance in the initial training phase, learning the initial set of ABA1 or ABB1 items. As can be seen in the figure, the SRN (unsurprisingly) had an easier time learning the final item of the ABB1 sequences (an adjacent dependency) than the ABA1 sequences (a nonadjacent dependency), though both networks rather quickly reached criterion performance of activating the correct final unit to an activation level of 1.0. Figure 20 shows the performance of the SRNs that were trained on the ABA2 items, as a function of the items on which they were trained in the first training phase. Figure 21 shows the performance of the SRNs that were trained on the ABB2 items, as a function of the items on which they were trained in the first training phase.

To determine whether the network’s learned significantly faster if its second phase of training was consistent with its first, a 2 x 2 repeated measures ANOVA was run on these models, with grammar type (ABA vs. ABB) and condition (consistent training in first and second training vs. inconsistent training in first and second training) and predictor variables, and with trial as a covariate. In this analysis, there was a significant main effect of grammar \( F(1,538) = 181.9, p < 0.001 \) and a significant main effect of consistency \( F(1,538) = 51.1, p < 0.001 \). There was also a significant interaction \( F(1,538) = 125.0, p < 0.001 \). Follow-up tests on this interaction revealed that while the effect of consistency was significant both for ABB-tested items \( F(1,538) = 10.9, p < 0.001 \) and ABA-tested items \( F(1,538) = 25.6, p < 0.001 \), that this
effect was much stronger for the ABA-tested items. This fact is also obvious from comparing the difference between the lines in Figure 20 and Figure 21.

Figure 19. Performance of the network on the first phase of learning, showing that the network learns the adjacent ABB dependencies more easily than the nonadjacent ABA dependency.

Figure 20. Performance of the networks that learned the ABA items in the second phase of training, as a function of which items they received in the first phase of learning.
These results demonstrate without ambiguity that SRNs can learn about abstract structure, and can transfer this knowledge from one set of items to another, even when there is absolutely no overlap in the items between training and test, and no microfeatures that the network can exploit. The network does show an asymmetry, with a much bigger consistency effect for the ABA items. This reflects the fact that the ABB networks are practically at ceiling the entire time, and that learning the new ABB2 items cannot go much faster as a consequence of what structure the network experienced in the first phase of learning. But the learning was significantly faster for the consistent, phase two training for both grammars, a clear demonstration that the SRN is learning the abstract structure of the first sequence, and is able to transfer that knowledge to the second set of items.

How was the network able to do this? Marcus’s description of the behavior of SRNs is correct in that the weights for the second set of A and B items are not being adjusted in ways conducive to applying the ABA or ABB structure to them, during the first phase of learning. But the weights are not being ignored, either. In fact, during the first phase of learning, the network is learning that these units are never used and never useful, and so it is adjusting all of their weights in a strongly negative direction. This is evident by noting that in Figures 20 and 21, the activation level for the novel items starts at 0, rather than at some higher base rate. The network has initially learned to turn these items off. This is why they cannot be used to immediately show transfer. The network starts phase 2 of training expecting these items to never occur.

How then, is the network learning and applying the abstract rule-like information? There is one set of weights in the SRN that Marcus and others have not discussed and where this information could be stored: the recurrent weights from the hidden layer that feed back to the

---

**Figure 21.** Performance of the networks that learned the ABB items in the second phase of training, as a function of which items they received in the first phase of learning.
hidden layer itself. If there is any place an SRN could possibly have learned something about the initial items’ structure that could transfer to the new items, it is there. These connections are where the network stores its representation about how its internal representations relate to each other. In the first phase of learning, these weights have been set so as to work well with the weights coming in from the input layer. In this sense, these weights are still bound (in that they are deliberately complimentary) to the weights coming in from the input items on which they were trained.

But because these weights have been adjusted and structured, they are now in a qualitatively different state than the initial, random state that they were in when the network began learning. In a sense, these weights have been set such that they compliment the item-specific weights of the input layer, representing the structural relations amongst those items. As a consequence, they are in a state that makes them more useful for representing the same structure in a new set of inputs, compared to a set of random weights. In the second phase of training, instead of needing to learn a mapping between the inputs to the hidden layer, the hidden layer to itself, and the hidden layer to the outputs, the network starts with the hidden layer in a state that is already “primed” and ready for the new items, if they occur in the same structural relations. All the network needs to do in the second phase is learn weights from the input layer to the hidden layer that allow the new items to usefully compliment the recurrent input, and learn to redirect the activation in the hidden layer to the new output items. In principle, the network could freeze the recurrent weights in the second phase of learning, not allowing them to be changed, and as long as the new input was structurally consistent with the initial input, the network would be able to learn weights for the new items that allowed it to perfectly predict these new items as well.

The lesson learned in this simulation fits very well with the broader theme in the other simulations. There are a number of ways to shortchange SRNs and other learning models, and leave them at a disadvantage relative to human learners. Some of the most important ways to do this are to make the situation ecologically invalid, taking out the cues that human learners are likely using in a given situation. As Seidenberg and Elman, and many of the others who have made distributed models of the ABA vs. ABB phenomenon rightly point out, there are a number of acoustic cues that infants may be using in these tasks, and adding those cues into the model allows the model to solve the learning problem. In this simulation we showed that SRNs can learn the abstract ABB or ABA structure, even without any of these distributed cues. But in order to exhibit this performance, the network needs to be given a different advantage that people also have: the ability to keep learning throughout the entire experience. People do not learn for a while, and then shut learning off while they get tested. People’s minds are constantly adjusting their representations as a consequence of changing input, and allowing networks to do this as well allows the networks to demonstrate knowledge that would be otherwise impossible to see.

This finding, that SRNs can learn to represent and transfer knowledge about abstract structural relationships, is a big deal. This is a demonstration that SRNs (and other connectionist models) are able to learn and represent the kind of abstract structure that many people feel is a hallmark of human language. What the SRN is learning is not dependent on any microfeatures of the stimuli, because there are none in the purely localist representations that were used. All that the SRN could possibly learn is something about the structural relationships that existed in the order of the items in the input. It learned that these items were occurring in either a repeating or

2 And in fact, a set of simulations freezing the weights in the second phase of training shows exactly this effect.
alternating structures, and it learned and stored that knowledge in a way that was generalizable to new inputs.

It should be noted, though, that this is not the same thing as the network actually learning a truly abstract rule, one that is completely independent of everything about the input. The network had to infer these abstract relationships from the input itself, and as a consequence the representations that were developed were still contingent on properties of the input. In this instance, while the networks were learning alternation or repetition, they were learning them as a function of six different A-items, and six different B-items. Because of this, that meant that the “alternation” situation (ABA) also involved learning a long distance dependency. The network didn’t just need to learn to switch back and forth between limited set of only two items. It needed to learn that after it switched from a specific A to a specific B, or that it needed to switch back to the same A that it had seen right before that B, rather than some other A. Due to the counterbalanced training, across the entire training set all six A’s were equally likely to immediately follow a given B. That was why learning alternation was more difficult in this case than learning repetition, which involved only adjacent dependencies.

Contrast this with another version of this model that only has four inputs and outputs (A, B, C, and D). This network might be trained on repetition (AAAA) or alternation (ABAB). In this example, because there is only one A and one B, alternation would be equally easy to learn as repetition, because both are based purely on adjacent dependencies. When this network is now switched to phase 2 of training (now it is given either CCCC or CDCD) it will show the same “consistency” transfer effects as our more complex network, and it does not show nearly the same asymmetry in difficulty as the original model (e.g. the transfer effect in the ABAB->CDCD condition is not much larger than the transfer effect in the AAAA->CCCC). In both the original model, and in this much simpler model, the network is learning abstract temporal relations and transferring them to new items. But in both of these situations, exactly what the network learns, how it learns, what it represents, and the transfer consequences thereof are still beholden to the exact distributional statistics in which the network learned the structural relations. Thus, this is still not pure “rule” learning. And given the actual data on human language learning and use, which argues against “pure” rule learning, this is about as close to rule learning as we want a model to get.

8. General Discussion

In the preceding five simulations, we have evaluated the Simple Recurrent Network (SRN) as a model of human learning in artificial grammar learning experiments. Specifically, we have examined its performance in simulating experiments involving learning of nonadjacent dependencies, in various circumstances where learning those nonadjacent dependencies appears to be relevant to important properties of language and how people learn it.

8.1. Summary of the Findings

In Simulation 1, we explored the basic performance of the SRN when it learns an AxB grammar involving a nonadjacent dependency where each A-element perfectly predicts each B-element. We showed that SRNs learn these nonadjacent dependencies, but we also showed that the networks, like humans (e.g. Cleeremans & McClelland, 1991), take a long time to do when the only cue to learning the dependency is the simple structural relationship. Further, Simulation
1 showed that SRNs transfer knowledge learned about A-B nonadjacent pairs to new strings containing novel intervening x’s, as people have been shown to do in a number of studies (Gómez & Maye, 2005; Willits, Lany, & Saffran, 2012).

In Simulation 2, we tested the SRN’s ability to learn nonadjacent dependencies in cases where the distance between the nonadjacent dependencies varies. We showed that when SRNs are independently trained on dependencies of different lengths, (as expected) the time that it takes the network to learn the dependency is a function of the distance between the items. Networks trained on nonadjacencies with three intervening items had a very difficult time learning the nonadjacency. However, we also showed that networks that are trained on dependencies with a mixture of distances separating them not only still learn, but also learn longer distance dependencies even better than networks trained on single span distances. Networks trained on a nonadjacent dependency that occurred at a variety of distances could transfer knowledge of that nonadjacency to distances on other than those on which it had been trained.

In Simulation 3, we looked at how variability in the set of items that intervene between nonadjacent dependencies affects a SRN’s ability to learn the dependencies. Specifically, we tested whether SRNs exhibit the characteristic u-shaped curve that people show, finding learning easiest when there is no variability or when there is extreme variability in the intervening items. We demonstrated that SRNs do show this u-shaped behavior under certain circumstances, in cases where they do not first get trapped in a local minimum whereby they only learn the adjacent dependencies in the input. The SRNs with the specific architecture and learning parameters that were used in this set of studies were more likely to get trapped in this local minimum as the variability increased, but a number of other, slightly more complex variants of the SRN show the u-shaped behavior without getting trapped learning only the adjacent dependencies.

In Simulation 4, we investigated how similarity structure, and its correlation with the nonadjacent structure that needs to be learned, affects learning of nonadjacencies. Newport & Aslin (2004) and Creel et al (2005) have shown that when nonadjacently related items are perceptually similar, this facilitates learning. Willits, Lany, and Saffran (2012) and Willits & Saffran (2011) have shown that this extends to semantic similarity, and does not require that items be similar, only that the similarity relations be consistent. In this simulation, we showed that SRNs also show these effects, with facilitated learning when the nonadjacent items are similar, and also when they are consistently from the two different categories (e.g. consistently pairing an animal with a food). Further, we showed that the dynamics of how this network learns demonstrates a number of other interesting effects that align well with the human data. One example is that the networks are likely to “false alarm” to a nonadjacent pair on which it had not been trained, if the item is from the same category as the correct item. Like people, the SRN was only likely to make these false alarms in cases where category similarity was correlated with the nonadjacent dependency.

In Simulation 5, we showed that SRNs (contrary to Marcus, 1998a, 1999b) can learn to represent abstract sequential structure and transfer this information to novel items. We showed that SRNs can demonstrate this effect even when they are given a purely localist input code, and there are no microfeatures other than the abstract sequential structure on which the network can rely to infer that knowledge. This was achieved by allowing the network to go through two learning stages, and by showing that while the network cannot immediately transfer knowledge
of the abstract structural relation to the new items, it is able to learn the new items more quickly if their structure is consistent with the structure in the first phase of training.

8.2. Evaluating the SRN as a model of human language learning

Overall, what do we make of SRNs as a model of human language learning? A number of characteristics have been attributed to language that make it special, and also that make it complex enough that simple associationist models cannot explain its nature or how it is learned. A few critical examples of these properties of language, and the SRN’s ability to model them, have been examined in previous work.

One such property is that language appears to be recursive. For example, one can embed sentences within sentences and still have the sentence be grammatical and sensible (e.g. *the man the boy liked left*). However, many have pointed out that even if language appears to follow rules that seem infinitely recursive, in fact people do not demonstrate this ability (Osgood, 1972, Christiansen & MacDonald, 2009). For example, adding one more embedding to the example above (e.g. “*the man the boy the girl saw liked left*”) is considerably more difficult to process and understand, arguably “breaking” the recursive abilities of the comprehension system. Recent surveys of actual language usage show that cases of three or more actual recursive embeddings in natural languages are exceedingly rare (de Vries, Christiansen, & Petersson, 2011). Christiansen and Chater (1999) explored how SRNs learn sequences with various types of recursive structure, showing that SRNs can learn simple recursive sequences but not more complex ones. Christiansen and MacDonald (2009) demonstrated that the model’s difficulties with various types of recursive structures are highly predictive of when people have difficulty with recursion. The conclusion is that while the ability to embed sentences within sentences is an interesting formal property of language, the ability of people to actually do this in practice is quite limited. Further, the SRN provides a good model of people’s limited abilities in this regard, and also their limitations.

A primary goal of this work was to systematically investigate a number of other “special” properties of language and analyze how well SRNs model those phenomena. In these simulations, we explored a number of ways in which nonadjacent dependencies – a critical aspect of language, and one responsible for a great deal of its complexity and expressive power – can be learned and represented by simple recurrent networks.

8.2.1 The Interaction of Similarity and Linguistic Structure. One major goal was to model the formal relationship between linguistic structures and how the learning of those structures interacts with the similarity of the items whose structure is being learned. Despite claims to the contrary (e.g. Chomsky, 1965), language is not a purely logical system of rule-based syntax operating over meaningless symbols. The units in language have perceptual and referential attributes that a learner can, and does, bring to bear when they are learning their language’s structure. The fact that the word *give* predicts that it will co-occur with words like *foods*, or the word *present*, is not an accident. It is the meaning of the words and the goals of referential communication that make it this way. The co-occurrences of nouns in sentences, the pairings of particular verbs with particular nouns, and of particular adjectives with nouns and adverbs with verbs are all, at some level, related to the referential meanings of those words.

The relationship between word meanings and linguistic structure is strong and useful. In natural language, the meanings of words and the distributional regularities of words are highly
correlated (Burgess & Lund, 1997; Jones & Mewhort, 2007; Willits, Amato, & MacDonald, 2012). Children can use these correlations to bootstrap learning in both directions – to learn about objects in the world given the distributional structure in which those words occur (Fisher, Gleitman, & Gleitman, 1994; Landau & Gleitman, 1984; Lany & Saffran, 2011), and also to learn about structural relations in language based on correlated semantic information (Willits & Saffran, 2012; Willits, Lany, & Saffran, 2012).

A comprehensive theory of language representation and learning will need to accommodate the fact that there is considerable redundancy between linguistic structure and world knowledge, that language learners exploit this redundancy during learning, and that this almost certainly has consequences for how this information is represented. In these simulations, we have shown that the SRN gives us a simple way to formally represent how similarity (whether perceptual or semantic) might influence the process of learning linguistic structure. We have shown that when an SRN has similarity structure encoded as part of a word’s representation, this has learning consequences for the SRN that very closely matches the learning consequences that have observed in children and adults. Thus far, the SRN and similar models appear to be good candidates for representing formal ways in which similarity and structural relationships interact during learning.

8.2.2. Learning Abstract and Rule-like Knowledge. To most linguists and number of psycholinguists, the fundamental nature of language is that it is a system of words, whose order is determined by a system of rules (Chomsky, 1965; Pinker, 1999). In English, the way in which a verb is changed from the present tense to the past tense is by the application of the “add –ed” rule (Berko, 1958; Brown & Bellugi, 1964; McNeil, 1976; Kuczaj, 1979; Marcus, 1992; Pinker & Prince, 1988). The definition of a sentence in English is that it is essentially a noun phrase followed by a verb phrase, with the option of recursively inserting mini sentences in the middle: i.e. S -> NP + (S) + VP (Bloomfield, 1932; Chomsky, 1957, 1964). Nonadjacent relations between parts of sentences (such as the fact that a subject noun and its verb have to agree in number) are stored as rules (Levelt, 1998; Bock & Eberhard, 1993). Semantic and syntactic information about words is stored in a rule-like way in the lexicon, such as that ditransitive verbs like give must have an agent, a patient, and a recipient (Dowty, 1979; Gruber, 1965; Jackendoff, 1990). To many, the rule-based nature of language is simply self-evident.

The resurgence of connectionism has led many, however, to question this characterization of language (Rumelhart & McClelland, 1986; Seidenberg & McClelland, 1989; Elman, 1991, 2009). In every case where there appears to be a rule at work, there often tend to be facts about the words’ phonology, orthography, distributional structure, and semantics, which suggest that rather than applying rule, people have instead learned a regularity in their environment that is inherently tied to the specifics of the words or other units of speech over which that regularity was induced. A number of connectionist models have been proposed to explain how a wide range of allegedly rule-based behavior is better explained by principles of statistical learning.

Proponents of rule-based approaches have typically responded to these models in three ways: (1) criticizing idiosyncratic specifics of the model implementations (e.g. Pinker & Prince, 1988, Marcus, 1999), (2) arguing that connectionist approaches cannot scale up to all the noise and complexity in actual language, or (3) arguing that connectionist models are missing the point, that they are merely models of language performance (mere implementations of rule-governed systems), not models of actual language competence, the underlying linguistic knowledge (Marcus, 1999; Fodor & Pylyshyn, 1988). Marcus (1998a, 1998b, 1999) provided a number of
examples that he argued were clear demonstrations of the rule-based nature of language and other aspects of higher cognition, and argued not just that connectionist models have not yet simulated these situations, but that due to their nature they are fundamentally incapable of doing so. Marcus’s arguments were really just an extension of earlier arguments against association-based systems, such as Bever, Fodor, and Garrett’s (1967) Terminal Meta Postulate, a formal argument for why association-based learning principles could never, in principle, learn anything abstract and independent of the items on which that knowledge was based.

Simulations 1, 2 and 5 were done to directly challenge these last two objections to connectionist models’ ability to learn and represent linguistic structure. Simulation 1 showed that SRNs that learn a nonadjacent dependency transfer knowledge of that nonadjacent dependency to novel contexts, with new intervening words. Simulation 2 showed that SRNs are robust to noise in the distance between nonadjacencies. SRNs are capable of modeling knowledge of nonadjacent relations, even when the usage of those relations is quite variable, as they are in natural language. Of the nonadjacent dependencies that occur at virtually every level of language, nearly all exhibit the property of being variable in terms of the distance between them. In English, articles (e.g. the) and possessive pronouns that precede nouns can be separated by a variable number of adjectives. Verbs and their direct objects can be separated by adverbs, preposition, the articles and adjectives referring to the direct object, and of course entire embedded clauses. Somehow these relations must be learned and represented, despite the variability that defines their actual usage.

It has long been argued that this variability is a deal-breaker for association-based and statistical learning approaches. Some have argued that, because it is self-evident that language is rule-governed, and that statistical learning and association-based approaches posit a fundamentally different representational structure, that therefore statistical learning approaches cannot possibly be a useful model of language learning and representation. But in Simulations 1 and 2, we have provided a simple demonstration that this is completely wrong. SRNs can infer an abstractable relationship between two nonadjacently related items, and transfer that relation to new situations with new intervening words, or even different numbers of intervening words. We showed not only that SRNs are robust to “noise” and variation in the environment in which these nonadjacent dependencies, but that in fact they thrive on this noise. The variability is critical to the SRN learning an abstract and transferable relation.

In Simulation 5, we made a similar and even more decisive demonstration of the fact that SRNs can represent rule-like information of the type that many think is critical to representing language. In the first phase of training, the SRNs learned that the input they were receiving was occurring in either an alternating (ABA) or repeating (ABB) pattern. We demonstrated that the SRNs had learned this abstract relation, by showing that in a second phase of training the SRN had an easier time learning a new sequence that was consistent with whichever pattern the network had seen before (either ABA or ABB). The SRNs that learned to do this did so using a purely localist representation and therefore had no microfeatures whose redundancy they could exploit. The only type of information the SRN could use to represent the pattern was the structural pattern itself. Nonetheless, the SRNs were able to this quite easily.

In sum, these simulations constitute important evidence that SRNs are capable of learning to represent many of the kinds of properties and relations that many consider to be a hallmark of human language. The networks learn a set of weights such that the network can make predictions about likely future states. But because these weights and activation states are derived from specific items and distributions of experience, the representations they develop are not properly
described as actual rules. Nonetheless, the networks are able to show the important benefits of having learned an abstract rule (such as learning knowledge of a relation that is invariant across distances or applicable to a new set of inputs). Demonstrating that the SRN can learn abstract and transferable structural patterns obviates many of the objections against statistical learning and association-based approaches. These findings also argue against the necessity of syntactically structured or rule-based representations.

8.3. Lessons Learned

It is one thing to show that a particular model (in this case, an SRN) can fit particular data or model particular behaviors. A model is even more valuable if it brings insight to the domain, insight that one would not already have had without the model. In these simulations, the way that SRNs solved these learning problems has taught us a number of new things about how people may be learning language.

One thing learned was an unusual consequence of why “more memory” helps with learning. In Simulation 1, we showed that an SRN with more hidden units learns more quickly. This wasn’t because the network needed more capacity to solve the problem. As we showed, the network only needed three or four hidden units in order to learn to represent two different nonadjacent dependencies. The reason more hidden units was helpful is because, since each hidden unit was contributing some activation to the output units, having more units meant that more weights were adjusted on each trial and therefore the overall rate of change of the network was faster. This is considerably different mechanism by which more resources contributes to faster learning, and one with some interesting consequences that could be explored in future simulations and experiments.

A second insight the SRNs provided is how and why similarity may bootstrap learning. Newport and Aslin (2004) argued that the reason perceptual similarity amongst nonadjacent items is helpful is because it creates a “pop out” effect or a Gestalt-like reorganization of input. This is certainly possible, though unproven. But it mechanism seems a little less likely to explain semantically similar effects such as those observed by Willits, Lany, & Saffran (2012), and it seems very unlikely as a mechanism for explaining the semantically “arbitrary but consistent” effects observed by Willits & Saffran (2012). The SRN provides an alternative account of this phenomenon. The SRN showed this facilitation effect because it was able to learn the category-level sequential structure more easily than the item-level sequential structure (i.e. it learning “Animal-BathroomItem-Animal” more easily than “doggy-towel-kitty”), and then when those two things were correlated, the category-level information facilitated the learning of the item-level information. It is very conceivable that this kind of a learning process might be helpful in a number of natural language circumstances. In the perceptual examples such as the stimuli used by Newport and Aslin, it is conceivable that people learn more general sequence structures first (such as CVCCVCV or CVCCVC, depending one’s native language), and then, when those patterns are correlated with actual phoneme nonadjacent dependencies, having this category level information would facilitate learning the items. Likewise, at the lexical level it’s easy to imagine that children learn something about characteristic sequences of grammatical categories (such as Subject-Verb-Object in English), and then are able to use the correlation of this structure with lexical categories (animate nouns are likely to be subjects, inanimate nouns are likely to be objects) to help learn the nonadjacent dependencies between specific words.
This point is related to the third lesson from these simulations. This lesson is that the nature of how SRNs learn sequential structure may have an effect of our conceptualization of “attention”, and how structure in the input might shift attention. A variant of the mechanism proposed by Newport and Aslin is that perceptual similarity shifts attention to the items that are similar, at the expense of those that are different, and allows one to more easily track the dependencies amongst those items even if they are not adjacent. The mechanism Gómez (1999) proposed to explain how variability in the intervening items affect learning of nonadjacent dependencies is similar: high variability in the intervening items leads one to effectively give up on learning those items, and instead focus attention on learning the only dependencies that one can, in this case the nonadjacent ones. The SRN, in the course of showing the same facilitation effects shown by the people in these experiments, suggests a different take on these events, one that does not involve invoking attentional shifts. Instead, the SRN is showing these effects because of distributional structure in the input. In the similarity case, it is the correlations between the easier to learn category information and the harder to learn item information that drives the effect. In the variability case, it isn’t that the network is “paying more attention” to slots one and three as slot two become more noisy. It is that, when slot two becomes more variable, this variability is a necessary byproduct of a shifting ratio of x to B items in the input. Attention is not being shifted from the X items to the B items. Each B item is just becoming more frequent relative to each X item, and as a consequence learning about the B items proceeds more quickly. The takeaway message is that in some cases, an affect that one might want to attribute to attentional differences might be more easily explained in terms of underlying and correlated distributional differences.

A final lesson from these simulations is one that was already implied from the behavioral work, and which is reinforced by finding the same effects in the models. Initial attempts to study nonadjacent dependency learning in people – whether for theoretically motivated reasons or just due to good-intentioned scientific reductionism – attempted to reduce the learning problem to its barest essentials. After doing so, researchers found that people had a very difficult time learning nonadjacent dependencies in the resultant artificial grammar learning experiments. In retrospect, this is perhaps not surprising. These experiments stripped out almost all of the cues that people were likely to be using to learn the structure. All of the experiments that have shown successful learning have all added cues back in. And importantly, these cues were all ones that are present, common even, in natural language. The same effect was found to be true time and time again in the simulations. SRNs are very powerful learning machines that, given enough time and resources, can learn just about any pattern. But like people, they learn them much more quickly when they are given some of the advantageous cues (such as similarity or variability) that surround us in the natural world.

8.4. Weaknesses & Future Directions

While the simulations have shown the SRN to be a good fit to human learning of nonadjacent dependencies in artificial grammar situations, there are a number of behavioral studies and further simulations that could be done in order to better understand the relationship between SRNs and human learning of linguistic phenomena, and number of future directions of research to make the present work more valuable and more interesting.

One example is the performance of the models in Simulation 3, involving the effect of variability in items coming between nonadjacent dependencies. This model was not a perfect fit
to the human data, due to the fact that the model was more likely to get stuck learning only adjacent dependencies as the set size increased, an effect that does not appear to be present in humans. Although it should be noted that there might have been individual differences in human performance showing that this is indeed the case, without access to the person-by-person data, this is impossible to know. But it is possible to imagine a scenario whereby some small number of participants are more likely to get trapped in the adjacent dependencies as the set size increased, but that this effect was swamped by much better performance on the larger set sizes by those that did not get stuck. Running a replication of this experiment with an eye on the individual differences data would be useful in this regard. However, even if this effect is not present in people, the reason SRNs demonstrate this behavior is clear and sensible, and other slightly more complex variants of connectionist models do not exhibit this tendency. This is evidence for the fact that while the SRN’s simplicity is useful for capturing and clearly explaining a number nonadjacency-learning phenomenon, a slightly more complex model is likely to be necessary to capture some of the nuances of the human data. Future work could perform a systematic investigation of some of these architectural and training differences, such as Sibley et al.’s Sequence Encoder (2010), or the more continuous methods of providing recurrent feedback (Pearlmutter, 1989, Allen & Seidenberg, 1999) to see which of these models scale up to more complex learning problems in a way that better fits what people are doing and provides insight into why.

Another way this work could be improved is by improving the match between the type of data that is being compared between the models and the people. In the models, we are analyzing the models’ learning trajectories; how fast did the models learn, and which conditions had qualitatively different performance. In the human data, we are usually interpreting a point estimate (performance at a particular point in time). People had fixed exposure period, and then were tested in some way. In some situations, comparing these two types of data is difficult. It is easy to imagine a situation where the condition that is performing better may flip over the course of a learning trajectory. How then, do we compare that to person’s performance at a single state in time? The ideal solution here is to redesign the experiments so that we are gathering continuous data over the course of learning, allowing us to compare the model’s learning trajectory to the people’s learning trajectory. A few experiments have done this by having the people perform online prediction or reaction tasks on an item-by-item basis, showing that reaction time decreases as participants learn the structure of the input (Cleeremans & McClelland, 1991; Onnis et al., 2012). Having this kind of data at our disposal for future attempts to model this kind of data would be extremely useful. Changing the task in this way could, however, also change what people are learning and representing, and this would need to be considered in future studies and models.

Another future consideration deals with how similarity was construed in Simulation 4. In this simulation, we used the simplest possible representation of similarity – having two items that were denoted as similar sharing a second node that was only active for members of that group. This is, of course, an extremely simplistic operationalization of similarity. We used this formalization because, in this first simulation, we merely wanted to make the point that any kind of similarity structure will help the SRN learn a nonadjacent dependency. However, real similarity structure is likely to be represented in a complex and distributed fashion, rather than as sharing a single node (McRae, de Sa, & Seidenberg, 1997; Rogers & McClelland, 2006; Smith, Shoben, & Rips, 1972). In future work it would be interesting to explore how different instantiations of similarity would affect learning. What if, rather than all members of a group
sharing a single node, they instead share some degree of overlap across a larger set of features. Remember that in our simulations, the primary reason that the similarity structure was helpful in the Category Match and Category Mismatch conditions was that the category-level structure was easier to learn. People first learned “Animal-BathroomObject-Animal”, which then helped them learn “Dog-x-Cat” and “Bird-x-Fish”. How would this story change as the distributed representation of similarity affects how easy it is to learn and track the higher-level relations? How could different distributions of items, the consistency of their similarity, and each item’s typicality, affect learning? These would be interesting questions to explore in a model and behavioral experiments.

Another final step forward for this work will be to try to connect it back to models and experiments involving even more naturalistic situations. Now that we are acquiring an understanding of the basic principles of how SRNs learn nonadjacent dependencies in these various crucial situations, we can compare this to models and experiments involving natural language. All of the simulations can be applied in this manner. Regarding Simulation 2 and the effects of variability in distance between nonadjacent items, one can perform corpus analyses of actual nonadjacent dependencies, and draw analogies to the model. Are dependencies whose distance is more variable in natural language more easily processed in language comprehension, and more quickly accessed in language production? Regarding the similarity bootstrapping effects, what types of similarity contribute to these affects? Does it matter if the semantic relationship is taxonomic or thematic, or similar vs. associated? Does it matter if the semantic similarity is derived from world knowledge (e.g. words whose referents’ share featural or contextual overlap in the world), versus derived from language (co-occurring and distributionally similar words)? Regarding the learning of abstract and “rule-like” phenomena, these models raise a number of interesting questions that could be followed up in analyses of natural language and experiments. As we showed in Simulation 5, the SRN is learning something that is abstract and rule-like, and applicable to new situations. But its representation of the rule-like information is still tied to the nature of the distributional input from which that rule-like knowledge was abstracted. One could examine a number of the rule-like situations in natural language, find similar distributional contingencies in cases where that rule appears to be used, and see whether experiments suggest that people’s use of the rule is tied to the distributional circumstances in which that rule tends to occur in natural language.

8.5 Summary

In this work, we have shown that the Simple Recurrent Network (SRN) can be used to simulate and better understand how people learn nonadjacent dependencies in a number of circumstances that are crucial to language acquisition. This includes modeling the learning of abstract and transferable knowledge of nonadjacent dependencies, knowledge that can be transferred to a large variety of novel situations, circumstances, and contexts. It also includes the modeling of how perceptual and semantic similarity might interact with the learning of structural relations in language.

These models, as well as the behavioral experiments on which they are based, demonstrate the power of simple statistical learning mechanisms. When these mechanisms are combined with the realistic, complex, and immensely useful structure that is present in the environment, we see that these mechanisms sufficient for explaining the learning and representation of many of language’s defining properties.
9. References


de Vries, Christiansen, & Petersson, 2011


