Bundle Memory:
A Computational Model of Reference Comprehension

Sommers, R.P., van Gils, T., Hagoort, P., Nieuwland, M.S.

Abstract

Discourse comprehension requires the capacity to store referents in memory and to subsequently retrieve them when appropriate. The linguistic, psycholinguistic and neuroscientific theories of reference comprehension can roughly be divided into two coarse-grained computational frameworks: connectionism and symbolism. We propose a computationally efficient and neurobiologically plausible computational model of reference comprehension that synthesises these two frameworks: the bundle memory model. We show that this model can perform reference comprehension by having it read sentences containing one or more referents and answer comprehension questions about them. By lesioning either the connectionist or the symbolic parts of the model we show that the frameworks complement each other and that combining them leads to capacities unavailable to either framework alone. Moreover, we trained several recurrent neural networks to perform reference comprehension, which were initially intended as simple control models. Unexpectedly, some of these models performed just as well as the bundle memory model and showed behaviours which many (psycho)linguists believe such models should not be able to do. Both the bundle memory model and the recurrent neural networks may provide fresh hypotheses on how the brain could perform reference comprehension.

1. Introduction

Language comprehension requires the capacity to associate words with their appropriate referents. When a child exclaims "Teddy bear!" while pointing at a particular plushy in a room full of household objects, the cognitive task of the listener is to associate the linguistic utterance to the object the child refers to. In this example the listener's task is not very cognitively demanding, as the physical context (and long-term memory) allows immediate referential disambiguation. All of the necessary information is given at the same time, and hence, short term memory is unnecessary to pick out the proper referent. Other times, however, referents are not immediately present and have been introduced earlier in time: they then need to be retrieved from short term memory. For example, when you read a sentence like "The child received the stuffed toy it desired", the pronoun "it" refers back to "the child". These so-called anaphoric references can also cross the boundaries of the sentence, requiring information contained in the larger discourse. Moreover, information need not be repeated exactly, as we can infer which referent is meant based on the semantic similarities of anaphor and antecedent. For instance, you may understand that the term "the stuffed toy" refers to the teddy bear mentioned in the first example utterance, despite the fact that no teddy bears are mentioned in the second sentence at all.

The central question that this paper addresses is: how could the brain encode, maintain and retrieve referents and their attributes in and from short term memory in the context of reference comprehension? In this paper we propose a computational model capable of performing anaphoric reference comprehension in a simplified artificial language.

1.1. Symbolism and Connectionism

There are two main competing computational modelling frameworks in the literature: symbolism and connectionism. The use of these terms can be problematic at times: both terms have multiple meanings in the literature; they are coarse-grained terms that gloss over many details; and they are not necessarily mutually exclusive, since one can implement a connectionist network on a symbolic computer and vice versa (Graves et al., 2014, 2016). Still, they are extremely useful for our present purposes, as they offer different complementary ways of thinking about computation (Holyoak & Hummel, 2000; Smolensky, 1990). As we shall see, each models reference comprehension in a substantially different way and is good at capturing different aspects of it. The model we shall propose offers one such synthesis.

With a symbolic system we shall mean any system that can use discrete symbols or variables that can be assigned arbitrary (semantic) attributes as values. An example of a symbol is the representation that gets stored in memory when we arbitrarily combine the attributes <is a builder> and <Rob>. The capacity to reason with dynamically composed symbols has become known as dynamic variable binding (Hadley, 2009; Marcus, 2001). We hold that such a system requires a separation of the (lexico-)semantics of the bound attributes from the compositional, productive syntax that instructs the binding procedure (Hummel, 2011; Hummel et al., 2004). To see why, consider that being a builder is not necessarily bound to Rob - Anna, Bob Dylan, and the little boy at the beach can be said to be builders too. Therefore, the mechanism which binds <is a builder> to <Anna>, <Bob Dylan> or {<little>, <boy>, <at the beach>} should be decoupled from the meaning of any of these terms which are stored in long term memory.

So far, the only known (non-biological) example of a symbolic system is the digital computer or von Neumann machine. Digital computers make use of a separation between the CPU - control systems that perform the computations (≈syntax) - and their two forms of addressable read/write memory - long term memory/hard drive (≈semantics) and working memory/RAM (for storing temporary variables). Together these systems can perform dynamic variable binding in ways we shall discuss later. Symbolism is often associated with GOFAI, classical Computational Theory of Mind, hand coded models, disembodied amodal semantics, LISP programs, logic- or rule-based systems, localist codes and innately learned rules. While these associations reflect genuine correlations - both historically and functionally - on our definition they are not necessary characteristics of symbolism.

With connectionism we mean any system that performs parallel, distributed and continuous computations by the spread of activation over (large) networks that contain all of the semantic information in their dense connections (Rogers & McClelland, 2014; Rumelhart, 1989). Modern examples of connectionism are the artificial neural networks of deep learning that are currently popular in machine learning and cognitive neuroscience (Doerig et al., 2023; Richards et al., 2019). These artificial neural networks usually do not separate memory and computation and therefore are thought to struggle with certain problems that require

dynamic variable binding (Fodor & Pylyshyn, 1988; Marcus, 1998, 2001). However, neuroscientific evidence is more in line with these connectionist systems. The neuroscientific literature suggests that semantic information is typically encoded in the brain in a highly distributed manner: neural representations are stored over neural assemblies containing many neurons (Averbeck et al., 2006; Pouget et al., 2000; Shamir, 2014; Thorpe, 1989), not unlike the representations of densely wired connectionist systems. Moreover, the retrieval of information from memory goes hand in hand with increased activation of the relevant cortical brain area in which said information is stored, strongly suggesting that the cortex does not separate memory and computation (Christophel et al., 2017; Fuster, Joaquin, 1997), as assumed by symbolic models.

Our definition of connectionism also includes certain types of Bayesian models. While Bayesianism and connectionism are not always seen as similar, there is considerable overlap (Griffiths et al., 2012). Importantly, certain influential Bayesian models of reference comprehension (e.g. Xu & Tenenbaum, 2007) are implemented as semantic networks that are likely to share the same problems connectionism has (Goodman et al., 2014). There are edge cases, for instance artificial neural networks that do separate memory and computation, either through learning or by design (e.g. Graves et al., 2016). For clarity, whenever we use the label connectionism, we shall typically not mean these edge cases. Consequently, not all models employed in deep learning today are subject to some of the problems we will discuss.

The psycholinguistic and cognitive science literature on cognition in general is ambivalent about its preference for either connectionism or symbolism. On the one hand, there are plenty of connectionist theories on the market: these are especially popular when it comes to modelling psychological processes that involve non-discrete, graded information, such as priming (Rogers & McClelland, 2014). On the other hand, in the past, symbolic ideas - such as a language of thought which is similar to predicate calculus (Fodor, 1975), or theories involving addressable read-write memory (e.g, Atkinson & Shiffrin, 1968) - were dominant (Rescorla, 2020). More recently, some cognitive scientists have argued for a re-evaluation of theories of language of thought (Frankland & Greene, 2019; Quilty-Dunn et al., 2022) and for the necessity of addressable read-write memory (Gallistel & King, 2009; Kumaran et al., 2016).

1.2. Linguistic, and psycholinguistic theories of reference contain aspects of both symbolism and connectionism

Theories of reference in linguistics typically favour symbolic models. In linguistics, anaphora are sometimes modelled in an approach called dynamic semantics (Nouwen et al., 2022), a family of theories in formal semantics which includes discourse representation theory (Kamp, 1981), file change semantics (Heim, 1982), discourse semantics (Seuren, 1985, 1994) and dynamic predicate logic (Groenendijk and Stokhof, 1991). This family of semantic theories works with a standard predicate calculus where referents consist of combinations of

predicates (or attributes). For instance, the noun phrase "the stuffed toy" is modelled by binding the predicates Stuffed(x), and Toy(x) to the variable x that stands for a particular referent. But because standard predicate calculus proved insufficient to model anaphora, dynamic semantics introduced a memory-like mechanism that can cross the boundaries of sentences in order to represent discourse referents (Heim, 1982; Kamp, 1981; Seuren, 1994). Instead, it considers referents as files or variables that are stored in separate (memory) states that are updated over time as new words and sentences come in. As larger portions of the discourse are incrementally parsed, these variables turn into larger and larger conjunctions of discourse-specific attributes (and relations) that together make up the referent, e.g. the stuffed toy referent additionally includes the earlier mentioned predicate of <teddy bear>.

One popular theory of reference in particular is the cue-based retrieval framework (Lewis et al., 2006; McElree et al., 2003; Parker et al., 2017), which is a combination of both connectionism and symbolism. In favour of connectionism, the theory suggests that the retrieval of stored referents requires a content-addressable retrieval process that works by comparing the semantic content of the incoming cue to the semantic content of the stored referents. Content-addressable memory is a parallel process where the semantic information of multiple referents can be accessed simultaneously - as opposed to a serial search through potential referents. This was hard for purely symbolic models, which used slow serial read/write memory access, but easy to do in connectionist systems where activation in e.g. a neural network can take many parallel paths to access information simultaneously (Hinton, 1984). In fact, the equation the cue-based retrieval theory uses is exactly that of a one-layer feedforward artificial neural network.

In favour of symbolism, however, this theory assumes that these referents can be stored and retrieved in short term memory in a discrete, one-shot manner. This strongly suggests an addressable read/write memory mechanism. Such a mechanism is standard fare in digital computers, but is more difficult to create in connectionist systems where memory and computation are typically not separated.

The combination of these linguistic, neuroscientific, and psycholinguistic insights leads us to a conundrum: on the one hand, the notion of referents as variables or files suggests the involvement of discrete representations or symbols that can be dynamically composed of bundles of attributes. In addition, addressable read-write memory suggests the need for separate memory and computation, as found in the digital computer. On the other hand, content-addressable memory; graded, non-discrete semantic information; and the neuroscientific evidence for distributed information processing, point out a need for connectionist/Bayesian semantic networks that perform computation by parallel, continuous activation over distributed networks. Reference comprehension is a cognitive function that requires aspects of both connectionism and symbolism. This makes the conundrum sketched above likely a false dichotomy - and reference comprehension a useful testbed in which we can try resolving the paradox.

1.3. Binding problem and Problem of Two

The difference between the two theories is best exemplified by two prominent problems in the literature: the binding problem, and its more difficult special case, the Problem of Two. The binding problem is a problem that many connectionism-inspired brain theories have difficulties with: it pertains to the question of how two (or more) attributes are appropriately bound together by neural firing of two (or more) distinct populations of neurons (Feldman, 2012; Greff et al., 2020; Malsburg, 1995; Treisman, 1996; Yu & Lau, 2023). For instance, in a sentence such as "The female guinea pig slept", the attributes <female>, <guinea pig> and <slept> must be bound together. However, since the information of these attributes is encoded by different populations of neurons, the question arises as to how these different attributes are combined into a single discrete representation of a referent. The binding problem is especially difficult in the context of reference comprehension where such referents can be stored, maintained and retrieved from short term memory.

The Problem of Two is related to the binding problem but then involves sentences containing multiple similar referents (Jackendoff, 2003; Marcus, 2001; Pollack, 1990). Take for instance the following sentence: "The male guinea pig ate, while the female guinea pig slept." Here, the listener needs to bind the <male>, <guinea pig> and <ate> attributes together and keep the resulting bundle of attributes distinct from the bundle of <female>, <guinea pig> and <slept> attributes. The difficulty of the Problem of Two rests on a combination of four constraints which are difficult to jointly satisfy: it evidently requires a way to bind attributes together and the ability for downstream areas to distinguish and control multiple objects with similar combinations of attributes. In addition, it needs to be able to deal with completely novel combinations of attributes, i.e., we can very rapidly understand and remember novel referents like "the guinea pig with a yellow top hat on his head looked ridiculous". Finally, the solution must be both computationally efficient and biologically plausible, i.e., the algorithm must be able to deal with large numbers of possible referents; and the solution must make use of mechanisms that have clear biological substrates.

While in principle capable of implementing any function, the main difficulty for connectionist systems is that their relatively static nature implies they lack the capacity of rapid variable creation and dynamic variable binding (Hadley, 2009). These properties are assumed to be necessary for dealing with the compositionality and productivity of language and thought (Fodor & Pylyshyn, 1988; Frankland & Greene, 2020; Marcus, 2001). The dynamic creation and binding of variables is the capacity of a system to combine multiple new combinations of features as variables and temporarily store them so that they may later be retrieved. Importantly, this creation and combination is done on the fly, and is therefore thought to require more degrees of freedom than can be afforded by connectionist systems (Hummel, 2011; van der Velde & de Kamps, 2006).

Besides incorporating symbolic mechanisms, connectionists have two strategies to solve this problem: the first strategy is to store all possible combinations of attributes in memory, and then retrieve them whenever prompted. This strategy leads to concerns of computational efficiency, as the compositionality and productivity of language implies a combinatorial explosion of the number of possible referents (and sentences) that have to be stored in memory (which we shall discuss more thoroughly in the discussion). The second strategy is to use forms of recurrence, which could then potentially learn how to dynamically store and retrieve the referents in the network without explicitly implementing forms of dynamic variable binding as inductive biases. However, neurobiologically plausible computational models show that sustained firing through recurrence is by itself not as stable a form of working memory as is needed to model human behaviour (Durstewitz et al., 2000; Lundqvist et al., 2018). Models that have longer lasting mechanisms such as spike-rate adaptation or NMDA receptor synaptic plasticity provide more stable memories (Fitz et al., 2020; X.-J. Wang, 1999). Moreover, existing recurrent neural networks often fail at establishing anaphoric relations between words in a sentence (Sorodoc et al., 2020), and do not succeed at generalising the compositionality required for it (Lake & Baroni, 2018; Loula et al., 2018). There are other problems with connectionist networks too: namely, their tendency to focus on statistically likely events makes it harder for these systems to deal well with violations of statistical correlations, for instance in cases with unlikely attribute combinations (Puebla et al., 2020). Consequently, concerns with computational efficiency, stability, compositionality and correlation violation threaten the usefulness of purely connectionist models of reference comprehension.

In contrast, symbolic theories offer an easy solution to these two problems, in much the same way as storing different files with similar content on your computer is not a particularly difficult problem. This is inherent to how computers work: information is stored in memory slots, and each slot of memory has an address. A control system allocates a referent/variable to a particular memory address, which can then store the referent's attributes in its dedicated memory slot.

For Problem of Two cases - cases with multiple referents that share the same information - computers use a memory efficient strategy: instead of unnecessarily copying the same information twice, computers store the attribute information just once. To distinguish the two referents, they then create two variables, one for each referent. The attributes of each referent are bound to the respective variables. Importantly, the variables do not contain the information itself but instead merely point to the memory address that does contain the attribute information. Thus, in the male and female guinea pig example, the <guinea pig> information is stored once, and the two symbols standing for the particular guinea pigs each point to that same attribute in memory (while each is at the same time pointing to different attributes that were not shared <male>, <female>). In doing so, these pointers essentially bind the information to the variable. Retrieval of the stored information is done by simply looking

at the location in memory the variables point at (an operation called "dereferencing"). Symbolic systems used this mechanism to implement the dynamic variable binding of attributes needed to solve the binding problem and Problem of Two (Hinton, 1984). Historically, this was how the programming language LISP was built, in which the famous reference comprehension model, SHDRLU was programmed (Winograd, 1972).

Digital computers can do this, partly because they are von Neumann machines, machines that separate memory from computation. If the brain were to separate memory from computation, it could likewise dynamically bind semantic information stored and processed in one location to (temporary) variables stored in other locations in memory. Yet there is a reason why symbolic theories, while popular in the early days of cognitive science and the computational theory of the mind, are currently less so: symbolic theories often made use of very rigid and discrete if-then statements to perform their tasks. This meant they failed to explain a range of other phenomena that required graded representations, e.g., priming effects, graded inference through similarity, and representing uncertainty (Haugeland, 1986; Loftus, 2007). Moreover, they made use of abstract symbols that were not grounded in sensorimotor modalities (Varela et al., 2016). Finally, for a long time symbolism failed to propose a convincing biologically plausible implementation (Eliasmith, 2013; Frankland & Greene, 2019), while connectionist and Bayesian models fared considerably better in these aspects. We shall revisit the issue of biological plausibility in the Discussion.

In sum, the two classes of theories and models are solutions to different subsets of a larger set of problems. We dub problems in this superset Symbolic (Re)combination, Retention and Retrieval (S3R) problems. S3R problems all involve our capacity to create and manipulate discrete representations (Symbolic) that are rapidly composed of (sometimes arbitrary) combinations of attributes (Recombination). Furthermore, these symbols can be safely retained in memory (Retention). We assume that these attributes can themselves be graded and distributed and thus have to be retrieved through content-addressable memory (Retrieval). Throughout the introduction so far, we have already discussed quite a few examples of this set of problems: the encoding, storage and retrieval of anaphoric referents, the binding problem, the Problem of Two, compositionality, discourse incrementation, inference, and correlation violation. We propose a computational model that can deal with the entire set of S3R problems, as each of the subtasks is necessary for good models of reference comprehension.

It should hopefully be clear that the subset of S3R problems that symbolic models solve seems to complement the subset of S3R which connectionism is able to solve. Therefore, we think it is likely that the problems in S3R share a common solution that can be reached by combining the two frameworks. This is not a new insight by any means (Harnad, 1990). The artificial intelligence (AI) literature contains a collection of attempts at synthesis of connectionism and symbolism. These are variously called symbol-connectionist hybrid models, neurosymbolic AI, or sometimes implementational connectionism. The latter is often

contrasted with eliminative connectionism, which argues that symbols are not needed to model cognition. Models that marry properties of both frameworks are not only restricted to the AI literature: within the domains of psychology and neuroscience, similar models and theories also exist, but they are not always connected to the larger symbolism/connectionism debate. Models and theories of this type are largely scattered throughout different parts of the literature, covering topics as diverse as working memory (Cowan, 2001; Fiebig et al., 2020; Manohar et al., 2019), language comprehension (Baggio and Hagoort, 2011), hippocampal/episodic memory (Kumaran et al., 2016; Meeter & Murre, 2005; Teyler & DiScenna, 1986), basal ganglia/thalamic gating (Hayworth & Marblestone, 2018; Kriete et al., 2013), and visual perception (Kahneman et al., 1992; Lades et al., 1993), as well as more general AI (Graves et al., 2016; Hadley & Hayward, 1997; Schmidhuber, 1992) and computational neuroscience models (Eliasmith et al., 2012; Hayworth, 2012; Smolensky, 1990). Despite the successful modelling of this large variety of phenomena within different domains, the similarities between these models and theories as well as their potential has likely not yet been fully recognized, explaining why they have not yet received a large amount of attention outside of select subfields. The bundle memory model we propose here shares many - but not all - of the characteristics of this family of models, but then applied to the specific problem of (anaphoric) reference comprehension.

1.4. High level description of bundle memory model

Roughly, our bundle memory model consists of three parts: a connectionist semantic network responsible for storing long-term knowledge, a symbolic bundle memory module providing a mechanism for dynamically combining and storing this long-term knowledge, and a control network which enables communication and coordination between these systems. The connectionist semantic network contains all of the long-term information about the attributes themselves as well as the relationships between attributes. For instance, in the network it would be stored that a dog is not a cat, that dogs bark, and that cats and dogs share some features, e.g. both are mammals. This also includes information of a less discrete, more probabilistic nature, such as the more continuous relationships between professions and clothing styles: e.g. a doctor is less likely - but still able - to wear a sweater.

The bundle memory module, as the name implies, is responsible for bundling the different attributes together, storing them and subsequently retrieving them. It acts as a memory buffer, and similar to a digital computer, contains multiple registers capable of assigning any single one to a possible referent. Each register is connected to each of the attributes of the semantic network, and by dynamically binding these attributes to the bundle/register using a fast Hebbian learning mechanism, the module is capable of binding any arbitrary combination of attributes together onto a particular referent. Subsequent reactivation of a bundle/register, either by the control network or by activating one of the attributes in the semantic network, then reactivates all of the connected attributes in the

semantic network through the backprojections from memory module to semantic network. Importantly, the module itself does not contain any information about the attributes: it merely binds pointers of the attributes to the bundle node according to the instructions of the control network. This implies that the representations of bundle nodes are not fixed and that the nodes can be re-used to bind referents with entirely different sets of attributes at later moments in time. The bundles/registers are kept distinct from the semantic network. This allows the bundle memory module to instantiate two tokens of the same type, by allocating each to their own register. Finally, only one register can be active at a time, ensuring that the two tokens are kept distinct and the two sets of attributes are not bound together even when both are present in the context. This makes the bundle memory module indispensable for Problem of Two cases.

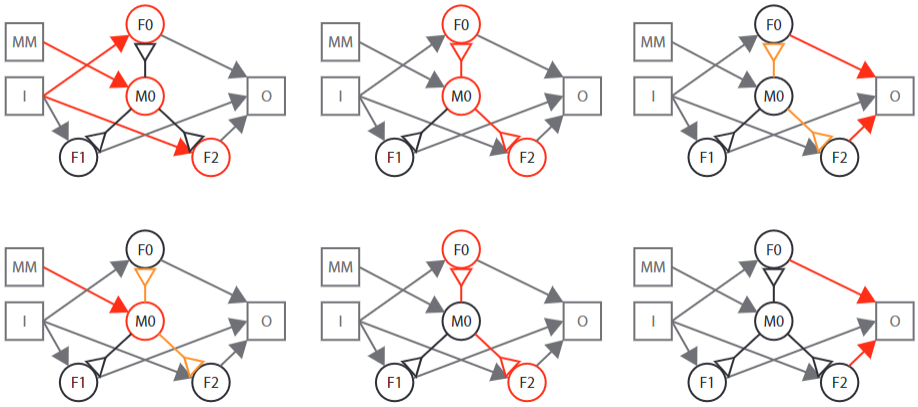


Figure 1. Diagram Depicting the Binding Mechanism in Action.
Top-left and top-middle: Input (I) and memory management (MM, part of the control network) result in concurrent activation of both the feature nodes (here, F0 and F2) and a single bundle node (M0).
Top-right: simultaneous activation of feature nodes and bundle node causes strengthening of activated synapses.
Bottom: subsequent activation of the bundle node by MM then reactivates both bound feature nodes which then project to output network (O).

The control network stores the (currently) hardwired rules that determine what instructions should be given to the bundle memory module, thereby controlling both it, and indirectly the semantic network as well. The control network acts as a filter and as memory management that prevents the bundle memory module from binding everything to everything else. It also enables controlled retrieval of particular referents. The instruction set currently includes a match and retrieve instruction, a store instruction and a query instruction. The memory operations given by the control network instruct the bundle memory module either to match incoming linguistic information with old referents to retrieve an old referent, or to set up a new referent representation.

The control subnetwork decides which operation to perform based on extrinsic syntactic cues in the linguistic input. The store and retrieve control instructions roughly correspond to a linguistic feature that is marked by most languages: definiteness (Aguilar-Guevara et al., 2019). In English, definiteness is marked by the different articles, “the”, and “a/an”. Indefinite

noun phrases ("I saw a clown today.") indicate the introduction of a new referent into a discourse, while definite noun phrases ("The clown wasn't very funny") are typically - though not always - used to denote already introduced referents (Aguilar-Guevara et al., 2019; Heim, 1982; Kamp, 1981; Seuren, 1985). The control subnetwork uses definiteness to decide whether to create a new bundle or whether to search for an existing bundle in memory.

We can use the metaphor of a webshop to illustrate how the different components of the bundle memory model jointly perform reference comprehension. Imagine you are a webshop application and are responsible for keeping track of the various orders (=referents) of your customers. Your long-term memory (=semantic network) contains a large selection of products (=attributes). The customers can use the webshop site to pick an arbitrary combination of products that jointly constitute their order (=binding problem). It is absolutely crucial for your job as a webshop that you do not mix up the orders of the customers, even though some of the orders might consist of the same type of items (=Problem of Two). To make your job easier, you assign to every customer a virtual shopping cart (=bundle in bundle memory module) that contains the products they have chosen. These do not really contain the products themselves, nor their full product information (=semantic content). Instead, it is sufficient to store a reduced representation (=symbol) and a link (=pointer) to the webpage that does contain all of the specific product information (=long term memory location). By listening to the instructions given by your customers - Henry wants to buy a box of chocolates, Sarah wants to buy the movie "Interstellar" - you can put the correct products in the correct carts. Interestingly, in many cases you (=control system) do not need to know exactly what these products are that they are buying, as long as you are following the protocol (=syntax) that regulates whose clicking behaviour belongs to whose shopping cart. But since there are correlations between products, you can use your long-term memory to predict to some extent what your customers will do (=graded inference). Similarly, when put together, the semantic network, the bundle memory module and the control system can perform reference comprehension.

We wish to test our bundle memory model on a set of subtasks involving reference comprehension. These subtasks are designed to test different subproblems of the problem set S3R (more detail in the Methods section below). We chose these subtasks based on the expectation that our bundle model could solve them. They do not reflect the complete set of requirements of reference comprehension (important limitations are considered in the Discussion section). However, they still serve as a proof of concept for a promising type of models going forward. To show the value they add, we compared the performance of our model against several control models: a couple of lesion models (a purely symbolic model and two Bayesian models) that reveal the added value of the hybrid approach; and two types of recurrent neural networks, recurrent neural networks (RNNs) and long-short-term memory networks (LSTMs) (Elman, 1990; Hochreiter & Schmidhuber, 1997; Jordan, 1997). We chose the latter networks to test how well classic connectionist models would generalise to new

referents (i.e., new combinations of attributes) when explicitly trained on our reference comprehension test suite (including the Problem of Two).

2. Methods

2.1. Task

In order to test and compare different models of reference comprehension, we constructed stimuli with a simple artificial language. This dataset consisted of 12 subtasks, each addressing a functional requirement a good model of reference comprehension needs to fulfil. Each item in the dataset is a sequence of words, typically starting with one or more declarative sentences and ending with a question the model is supposed to answer. For instance, one of the simpler tests involved a sequence like "There is a man with a hat. Does the man(/woman) have a hat(/bag)?"

2.2. Stimuli creation

Each subtask has a number of conditions, with the number of conditions dependent on the specific test (147 conditions spread unevenly across the 12 tests). The different conditions were created by varying select words in the sequence, e.g., in the above sequence when we replace "hat" by "bag" in the question only, the expected answer changes. Not all conditions are of equal theoretical interest. Some merely served as control conditions that vary the order in which information is presented or as fillers to ensure sufficient variation exists in the data set. This was especially important to avoid overfitting the RNN and LSTM control models on only single word orders.

Every condition has N distinct items, where an item consists of a sequence of sentences. We varied the number of items per condition (N=100/1000/10,000) when training the RNNs and LSTMs in order to determine whether increasing dataset size could lead to better models. Accuracy did indeed increase with the number of items per condition (see Figure A1.). The items were generated from manually created templates specific to each test condition, where some of the content words were filled in automatically. For instance, in the example sequence above, the words "man", "hat", "woman" and "bag" could be replaced by a combination of different words, taken from a specified word class (see below). By sampling different combinations of content words and filling these into the template, we could generate many different items. Our largest dataset consisted of 1,470,000 unique items (10,000 x 147 conditions). These datasets were split into training (60%), test (20%) and validation sets (20%).

The content words used in the sequences were also artificially generated. We defined 26 different word classes. Twenty-five of these word classes were words for particular attributes. These word classes differed in two respects: 1) in the conditional probabilities with respect to a particular category, i.e., P(scientist | lab coat), and 2) in relative word probability/frequency,

i.e., P(lab coat). We defined five uniform probability distributions with different likelihoods (neutral/negligible, low, medium, high, very high). For each word, we sampled a different conditional probability and a different relative word probability. Thus, resulting in 5 x 5 = 25 different word classes. The final, 26th word class contained the words for the different categories, e.g., man, woman, scientist, lawyer. We created 7 categories, and therefore defined 7 category words, which were spread across 3 independent dimensions, e.g., gender, occupation. For each category, we created four words per attribute word class, resulting in 700 different attribute words (7 x 4 x 25). This brings the total number of content words up to 707. Note that for simplification we assume that no lying or misrepresentation is possible: words perfectly predict the categories and attributes that a referent possesses, i.e., p(attribute | word) = p(attribute) = p(word).

The sequences that enter our model consist of artificially created words, but to improve readability and emphasise the parallels with natural language, we report them here in a natural language form. Each of the sequences in this test suite is provided as input to our model, with model resets in between different sequences.

2.3. Subtasks

We will now describe the different subtasks and their different conditions. In the example, different conditions are indicated by parentheses.

2.3.1. One Referent

Example sequence:

Discourse: There is a man with a hat.

Question: Does the man(/woman) have a hat(/bag)?

Answer: Yes(/No/Maybe)

A model that can do this subtask can store a single referent (in short term memory) and subsequently retrieve it (through content-addressable memory) to answer a query. This subtask is the simplest referential task we could think of. Different conditions consist of different content words in the question. These vary whether the noun anaphorically refers back to the discourse referent or not and whether the question is true or false.

2.3.2. One Referent Multiple Attributes

Example sequence:

Discourse: There is a man with a hat and a bag.

Question: Does the man(/woman) have a hat(/bag/bike)?

Answer: Yes(/No/Maybe)

A model that can do this subtask can store a single referent that possesses multiple attributes (in short term memory) and subsequently retrieve it by activating the content of any one of its attributes. Like the One Referent subtask, different conditions contain different content words in the question, varying anaphoric reference of the question's subject as well as its answer.

2.3.3. Discourse Incrementation

Example sequence:

Discourse: John is a man. The man has a hat. The person with the hat has a bag.

Question: Does John have a bag(/hat/bike/Is John a man)?

Answer: Yes(/No/Maybe)

A model that can do this subtask has a rudimentary capacity of incrementing a small one-referent discourse. It is quite rare for two related attributes to enter the sensory system exactly at the same time. During comprehension of a (single) discourse, this never happens. Thus, reference comprehension is an incremental business, proceeding one word, phrase and sentence at a time. A good model of reference comprehension must therefore be capable of binding together temporally extended attributes. Note the similarity of this task to transitive inference of the form $A \rightarrow B$, $B \rightarrow C$, thus $A \rightarrow C$ (Lazareva, 2012). Different conditions request information about different attributes. These different attributes should either have been

bound to John at different times in the discourse, or they are entirely new attributes that should not have been bound to John at all.

2.3.4. Two Referents

Example sequence:

Discourse: There is a man with a hat. There is a woman with a bag.

Question: Does the man(/woman) have a hat(/bag)?

Answer: Yes(/No/Maybe)

A model that can do this task can store and retrieve two referents with distinct attributes. This tests whether the model can correctly assign the correct attributes to the right referent and then correctly answer a question about either one. Different conditions vary which referent is referred back to and which attribute is requested.

2.3.5. Three Referents

Example sequence:

Discourse: There is a man with a hat. There is a woman with a bag. There is a boy with a bike.

Question: Does the man(/woman/boy) have a hat(/bag/bike)?

Answer: Yes(/No/Maybe)

A model that can do this task can store and retrieve three referents with distinct attributes. This extends the number of referents to three. Different conditions vary which referent is referred back to and which attribute is requested.

2.3.6. Problem of Two

Example sequence:

Discourse: There is a man with a hat and a man with a bag. The man with the hat has a bike.

Question: Does the man with the hat(/bag) have a bike(hat/bag/jacket)?

Answer: Yes(/No/Maybe)

A model that can do this task can store and retrieve two similar referents (same nouns). This task is supposed to be extremely difficult for connectionist networks to pass in a computationally efficient manner, and is supposed to test a model's ability to perform dynamic variable binding. Different conditions vary in terms of which referent the question is about and which attribute it is supposed to have.

2.3.7. Problem of Two Multiple Attributes

Example sequence:

Discourse: There is a man with a hat(/bag) and a bag(/hat), and a man with a hat(/bike) and a bike(/hat).

Question: Does the man with the bag(/bike) have a hat(/bag/bike/jacket)?

Answer: Yes(/No/Maybe)

A model that can do this task can store and retrieve two similar referents with multiple attributes. This is a harder version of the standard Problem of Two, forcing a model to store more attributes. Moreover, in some of the conditions, the two referents are even more similar to one another, sharing both the category (man) and a single attribute (hat) and only differing on one attribute (bag/bike). The increased number of attributes should decrease the likelihood that the recurrent neural networks can succeed by memorising all solutions. Different conditions vary in terms of which referent the question is about and which attribute it is supposed to have.

2.3.8. Category Inference

Example sequence:

Discourse: Robin wears a lab coat(/suit/shirt).

Question: Is Robin a scientist(/lawyer)?

Answer: Yes(/No/Maybe)

A model that can do this task can infer a category from a highly correlated attribute. Due to their discrete if-then rules, symbolic models typically struggle with graded inferential processing/association. Different conditions vary the relationship of the attributes in the discourse to the category in the question. Either these attributes are correlated, anti-correlated or uncorrelated.

2.3.9. Attribute Inference

Example sequence:

Discourse: Robin is a scientist(/lawyer).

Question: Does Robin wear a lab coat(/glasses/wig/suit/shirt)?

Answer: Yes(/No/Maybe)

A model that can do this task can infer an attribute when given a category: the reverse of category inference. Different conditions vary the relationship of the categories in the discourse to the attributes in the question. Either these attributes are correlated, anti-correlated or uncorrelated. We also varied the base rate: that is, infrequent attributes (lab coats/wigs) are less likely to be worn than frequent attributes (glasses/suits), even though both types of attributes are correlated to scientists/lawyers.

2.3.10. Six Questions

Example sequence:

Discourse: There is a man with a hat.
Question 1: Is the man(/scientist/person with a hat/bag) a person with a hat(/bag/scientist)?
Answer: Yes(/No/Maybe)
Question 2: Is the person with a hat(/bag/scientist/man) a scientist(/person with a hat/bag/man)?
Answer: Maybe(/Yes/No)
⋮
Question 6: Is the man(/scientist/person with a hat/bag) a scientist(/person with a hat/bag)?
Answer: Maybe(/Yes/No)
A model that can do this subtask can consistently answer multiple questions about the same referent. This subtask is primarily included to challenge the LSTM and RNN models to store information for longer durations, which is something especially RNNs are known to struggle with (Bengio et al., 1994; Pascanu et al., 2013). Different conditions of this subtask used different permutations of the 6 questions. The 6 questions are quasi-randomly chosen out of the 12 possible questions. In some conditions, the questions require graded inference (either category or attribute inference).

2.3.11. Correlation Violation
Example sequence:
Discourse: There is a lawyer(/scientist) wearing a lab coat(/glasses/suit/wig/shirt).
Question 1: Does the lawyer(/scientist) wear a lab coat(/glasses/suit/wig/shirt)?
Answer: Yes(/No/Maybe)
Question 2: Does a lawyer(/scientist) wear a lab coat(/glasses/suit/wig/shirt)?
Answer: No(/Yes/Maybe)
A model that can do this subtask can overcome statistical correlations through an updated discourse model, i.e., it should say that scientists typically do not wear wigs (Question 2), but when explicitly told that there is a scientist who wears one, it should be able to store and retrieve this information (Question 1). Puebla et al. (2020) have shown that connectionist models struggle with sudden violations of learned statistical correlations. Symbolic models can do this as they can bind arbitrary - and thus also anti-correlated - attributes. This subtask further tests these ideas. Like the Attribute Inference subtask, different conditions vary the relationship between attribute and category (correlated/anti-correlated/uncorrelated) and its frequency (infrequent/frequent).

2.3.12. Problem of Two Category Inference

Example sequence:

Discourse: There is a man with a hat and a man with a bag. The man with the hat(/bag) has a lab coat(/lab flask/suit/bike). The man with the bag(/hat) has a suit(lab flask/lab coat/bike).

Question: Is the man with the hat(/bag) a scientist?

Answer: Yes(/No/Maybe)

A model that can do this subtask can solve our most complicated Problem of Two cases. The model has to store two similar entities and from their multiple assigned attributes infer whether a queried referent belongs to a particular category or not. This subtask essentially combines two subtasks: the Problem of Two Multiple Attributes, and the Category Inference subtasks. The subtask is included to prevent the LSTM and RNN control models from overfitting only on the vanilla Problem of Two subtask. That subtask could be less memory-intensive as there the networks were not forced to store multiple combinations of attributes nor to retain inferential capabilities. Different conditions vary which referent is referred to in the third and in the fourth sentence of the discourse. In this way, the order in which the third attribute gets assigned to the referents is controlled, which prevents learning a positional code. In addition, different conditions vary which referent gets referred to in the question. Finally, the type of attribute-category relationship is manipulated by assigning different attributes to the referents (correlated/anti-correlated/uncorrelated).

2.4. Computational explanation of bundle memory model

In the introduction we divided our model into three components: a semantic network, a bundle memory module and a control system. We further subdivide our semantic network into an output and input unit and get the four processing units seen in Figure 2. The input and output units can both be considered part of the same hierarchical generative model for semantic knowledge.

We will first elaborate on each of these four units (following the operations described in Figure 2.), describing the formal requirements in order to be able to accomplish specific tasks, like reference resolution and the Problem of Two. This mid-level description should generalise beyond our set of tasks and our specific implementation, and is intended to provide a general framework for incorporating bundle memory into different types of computational models, e.g. Bayesian, connectionist, spiking neural networks, etc. Subsequently, we will elaborate on our implementation of this mid-level model, as we aim to provide a minimal implementation that meets the functional specifications. Broadly speaking, our aim is to provide a proof of concept that this mechanism can explain reference, and we have hence kept this model as simple as possible.

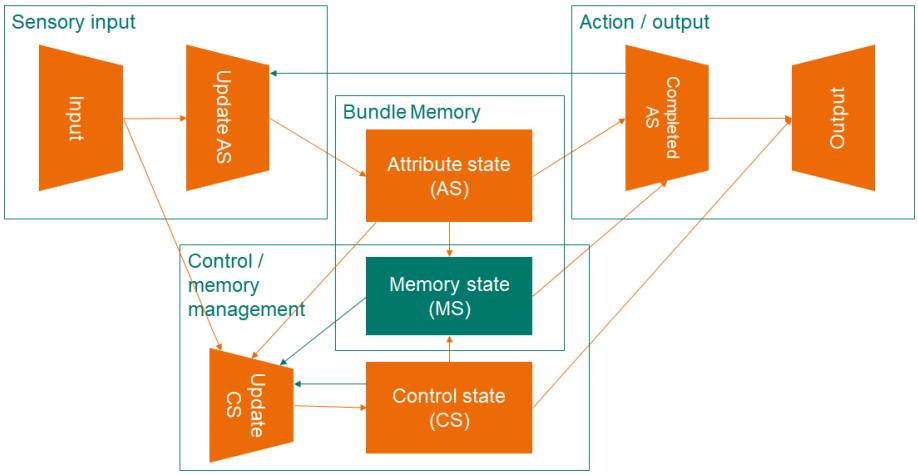


Figure 2. A Diagram of the Four Main Processing Units.

Their internal operations are indicated by orange and green blocks. The flow of information of these operations is indicated by arrows.

2.4.1. Input Unit

When hearing a sentence, information about the semantic contents and the expected response must be extracted. The input unit (IN) receives sensory input and extracts task-relevant attribute (AI) and control (CI) information, and subsequently combines this attribute information (AI) with the current attribute state (AS) of the network:

(1) Extract input: $IN \rightarrow \{CI, AI\}$
(2) Update state: $\{AS_{t-1}, AI\} \rightarrow AS_t$

For example, while listening to a sentence about a man with a hat (IN), the information may be split into the control information (CI), signalling that this input contains information about a referent, and the attribute information (AI), about a hat and a man wearing this hat. This attribute information is integrated with existing attribute state (AS) information in the semantic network, if any; for example, some other information about this same man that was presented immediately prior. This processing step allows the most recent attribute state to be subsequently used in an appropriate manner, depending on the relevant control signals.

2.4.2. Control system

Given the information about the expected response, a course of action must be determined, taking into account existing memory information – does the task involve an existing referent, or does it introduce a new one? The control unit retrieves information about the state of the bundle memory module (MI), based on the previous memory state (MS) and current attribute state (AS). Subsequently, it updates the control state (CS) of the network using this memory information (MI), the task-relevant information (CI), the current attribute state (AS), and the current control state (CS).

(3) Check memory state: $\{MS_{t-1}, AS_t\} \rightarrow MI$
(4) Update control state: $\{MI, CI, AS_t, CS_{t-1}\} \rightarrow CS_t$

Using the same example input as above, the information about a man with a hat (AS) could be used to determine whether this information matches an existing referent that is currently kept in working memory (MS), resulting in information (MI) about what referent the current attribute state refers to, or whether it introduces a new (or ambiguous) referent. Subsequently, all this information can be combined to determine the currently appropriate course of action (CS): for instance, that new information should be stored together with an existing referent, that a new referent should be stored in working memory, or that further cortical processing is required.

2.4.3. Bundle memory module

Having determined what course of action to take, other information may need to be retrieved from memory to complete the intended task. The Bundle Memory unit combines the new attribute and control states (AS and CS) with the previous memory state (MS) to create, update, or retrieve attribute information (MAI) in bundle memory.

(5) Update/retrieve memory state: $\{AS_t , CS_t , MS_{t-1}\} \rightarrow \{MS_t , MAI\}$

This will actually execute any memory-related actions as determined by the control state (CS), for example retrieving an existing referent matching the input about a man with a hat, encoding new attributes into an existing referent, creating a new referent, or preparing to contrast a specific memory to the current attribute state, e.g. to respond to a question.

2.4.4. Output unit

Having updated and/or retrieved the involved memories and determined a course of action, the appropriate response or output must then be determined. This could involve contrasting a retrieved memory to the current memory state, or any other process involving either the attribute state or the retrieved memory information. The output unit uses the new attribute state (AS) with any attribute information retrieved from memory (MAI) and the current control state (CS) to update the attribute state (AS), and subsequently combines this new attribute state (AS) with the same information (MAI and CS) to determine the appropriate output (OUT).

(6) Update attribute state: $\{MAI, AS_t, CS_t\} \rightarrow AS_t$

(7) Determine output: $\{MAI, AS_t , CS_t\} \rightarrow \{OUT\}$

Again, using the same example input as above, any existing referents may be retrieved (MAI) and merged with the current attribute state (AS), e.g. that the man with a hat also has glasses. Alternatively, for a question (as specified in the control state (CS)), the attribute state (AS) may not be updated at all, but instead used to contrast any queried attributes (e.g. beard) with the retrieved memory (MAI), to answer the question about whether this particular man has a beard (OUT).

2.5. Implementation of bundle memory model

In our current model, equations 1-7 are implemented in a way that minimally meets the functional specification, to provide a proof of concept that this mechanism can explain reference. Each of these components could be replaced by a more complex implementation, e.g. a feed-forward neural network, recurrent neural network, or Bayesian network, and it is likely that such implementations would more closely mimic human behaviour. Here,

however, we aim primarily to qualitatively show that Bundle Memory can explain certain observable processes, like reference, without needing to deal with complete syntactic and semantic processing. The task and semantic network used in our model, as well as the functional implementation of each equation, is specified below.

2.5.1. Bayesian semantic network

We used a hierarchical generative Bayesian network to implement the semantic network. To be able to test graded inferential capabilities, we created categories that stood in probabilistic relationships to some of the attributes, e.g., the category "scientist" could be inferred from attributes like "wears lab coat". We constructed the semantic network by defining an underlying probability distribution. We used three category dimensions (two binary, e.g. scientist/not-scientist; one tertiary, e.g. introvert/extrovert/neutral). For each attribute, we specified its probability of being the case given the value of the category: $p(attribute\ a \mid category\ c)$. In order to create uncorrelated attributes, we made sure the attributes correlated only with one category dimension and were independent from the other two.

This Bayesian semantic network was the same as the one we used to generate the stimuli. The Bayesian semantic network had perfect information and thus any ANN trained to approximate the semantic information would not be able to do better than the Bayesian variant.

2.5.2. Bundle memory module

We implemented bundle memory by mimicking the neural structure depicted in Figure 1., where each attribute node can be bound to a memory node. In our simulations, we used 4 memory nodes, and encoded the binding strength information in an n × m matrix B, where n is the number of attributes and m the number of memory nodes. For retrieval, we compared the attribute state vector **a** (of size n) with the contents of the binding matrix using matrix multiplication: **a**B = **s**. The result is a vector **s** of size m representing the similarities between the attribute state vector and the contents of each of the bundle nodes. The module then determines whether there is a node that matches and whether that node is not ambiguous with another node This is done by checking if there is a matching node while all other potential matches are below the following threshold $0.8 \cdot similarity_{max\ match} - 0.5$. If these conditions are fulfilled, it selects the most similar memory node using $argmax(\mathbf{s})$. If not, it returns a signal to the control system. For binding new information, given an attribute state vector **a** and a memory node index *i* (which needs to be allocated by the control system), the bundle memory module can simply add and normalise (using a tanh activation function) the value of **a** to column i in the binding matrix B.

2.5.3. Extract input

The input is specified as a single symbolic token, either a word or a command signal. The words that are allowed as input match the concepts specified by our semantic network, and no distinction is made between nouns or adjectives. The allowed commands are *AskNew (RQ)*, *AddNew (MR)*, *AskGiven (MQ)*, *AddGiven (M)*, *Verb (V)*, and *End (. or ?)*, with abbreviations which may be used throughout this paper specified in between parentheses. Given a command other than a word, Verb or End, the current attribute state (AS) will be cleared. This is to ensure that the previous referent information will be cleared at the start of a new sentence, to prevent the conflation of features. The four initial commands, AskNew, AddNew, AskGiven and AddGiven, could each be split up in two components: task information, about whether the sentence specifies something or asks for something, and reference information, about whether the task regards a prior referent or introduces a new one. This information can generally be extracted from the syntactic or pragmatic cues in an utterance, which we have here simplified in the form of these four commands. The Verb command appears in place of the finite verb of a sentence, or is inserted after the subject in question sentences where the finite verb appears first, and this command signals a switch from matching or retrieval behaviour to recording or querying behaviour.

When a word is detected as input, it is forwarded to the attribute state update function, and the control state update function receives no command. When a command is detected as input, the attribute state update function receives no input, while the control state function is forwarded the given command. The exact specification for what is done with each command is specified in the control state section below.

2.5.4. Update state

The attribute state (AS) is specified as a set of attributes, where each attribute is one of the possible attributes in the semantic network. When updating the attribute state, the attribute associated with the current word input, if any, is simply added to the attribute state set.

2.5.5. Update control state

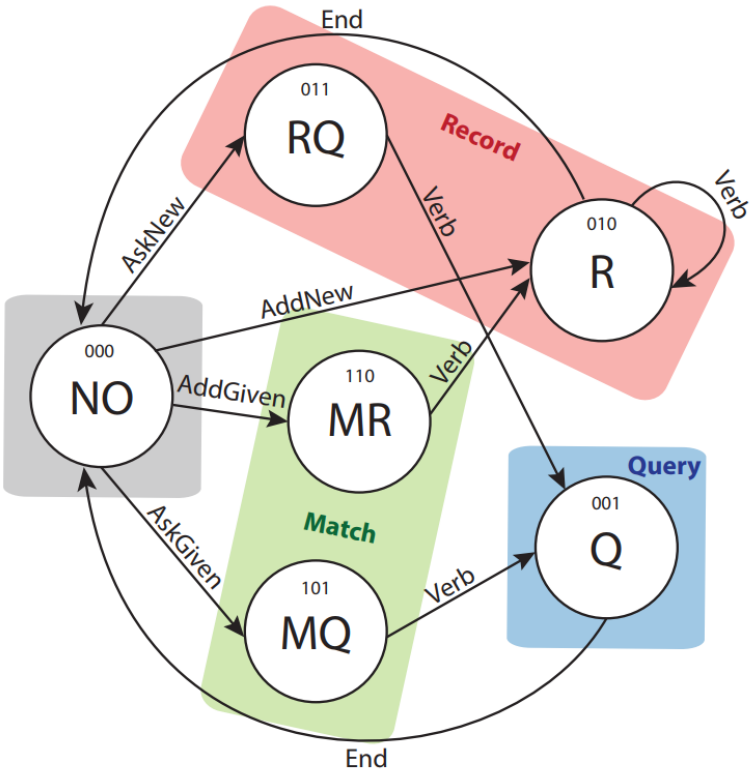


Figure 3. A Diagram of the (Deterministic) Markov Chain Describing the Control Flow of the Control State (CS).

Nodes describe the possible states, and edges define how this state changes depending on incoming commands. The shaded areas determine the action to be taken by the control area on the bundle memory module. Possible actions are to either perform no action (NO), to try to *Match (M)* an existing memory node, to *Record (R)* new information to a new or existing memory node (as specified by the current memory address), or to *Query (Q)* information from an existing memory node and compare it to the current attribute state (AS).

The Control State (CS) is defined as the combination of (1) a memory address register and (2) the current state of a Markov Chain, as displayed in Figure 3., with incoming commands altering the state according to this diagram. The current state is then used to determine the subsequent course of action, either to:

1. *Match* an existing memory node given the current attribute state (AS) and to update the memory address register to the best matching memory node, or to
2. *Record* new information to the memory node currently referred to in the address register, which could be a previously matched existing memory node or a newly allocated one, or to
3. *Query* the information in the memory node currently referred to in the address register, preparing the model to subsequently prepare it to the current attribute state (AS).

2.5.6. Update/retrieve memory state

Given the most recent attribute state (AS) and control state (CS) along with the current memory state (MS) of the bundle memory module, the model can now return the features of the indicated memory node on a Match (MR or MQ) or Query (Q) signal, and create or update the indicated memory node using the current attribute state (AS) on a Record (RQ or R) signal.

2.5.7. Update feature state

On a Record (RQ or R) signal, the retrieved memory state (MAI) will be merged with the current attribute state (AS):

$$AS_i \leftarrow AS_i \cup MAI$$

2.5.8. Determine output

Finally, the new attribute state (AS) and memory information (MAI) will be used to answer the question whether the current referent as stored in the memory node (MAI) is likely to have the attributes as described in the attribute state (AS), using the Bayesian Network described earlier:

$$\text{Query response: } p(AS \mid MAI)$$

2.6. Control models

Besides our own bundle memory model, we trained two types of recurrent neural networks to perform reference comprehension. These consisted of a single word embedding layer followed by either an LSTM (long-short term memory) or a simple RNN (recurrent neural network) layer and finally a dense output layer with three units (yes/no/maybe) and softmax activation function. We varied the sizes of both layers simultaneously (10/100/1000 units) to see whether network size had an effect on accuracy. As expected, accuracy did increase with network size (see Figure A1.). The other training parameters were fixed: we used a batch size of 2048; a Glorot uniform kernel initializer; an orthogonal recurrent initializer; a zero-bias initializer; no dropout; a clipnorm of 0.01; Adam optimizer; categorical cross entropy loss function; and used an early stopping procedure that stopped if the validation query loss did not change for 100 epochs, after which the model that scored best on the validation query loss was selected.

We also prepared three "lesioned" versions of our bundle memory model, in which parts of our network were removed. These serve as simple control models. In one such model, we lesioned the Bayesian semantic network of the model - resulting in a purely symbolic system we dub Overly Symbolic. The Overly Symbolic model can perform addressable read-write memory operations and can thus create distinct variables for different referents, but it has no way to model graded information. For the other two models we lesioned the (symbolic)

bundle memory module - since this can be done in multiple ways, we made two slightly different connectionist systems, the Associative Bayesian and Amnesic Bayesian. The Associative Bayesian accumulates and retains all the information encountered during the sentence, but it has only a single semantic network, and therefore cannot keep multiple referents apart; the Amnesic Bayesian resets the activation state of the semantic network every time it encounters a new word, thus forgetting what was said.

3. Results

3.1. The bundle memory model can perform all subtasks

We built the bundle memory model by combining the capacities of two complementary computational frameworks: symbolism and connectionism. In doing so, we hoped our model would be able to solve all of the problems in the S3R problem set, which consists of subtasks necessary for reference comprehension.

As expected, our model performed with perfect accuracy at all the subtasks we prepared (see Figure 4.). This includes symbolic subtasks: we show our model can parse a sentence word for word and successfully bind one or more arbitrary attribute(s) to a referent, maintain this information in memory, and retrieve it to answer one or more question(s) (One Referent, One Referent Multiple Attributes, Six Questions, Figure 5.5.). Our model could thus form a solution to the binding problem. In addition, when the discourse grows, the model can incrementally add attributes to the referent - even when this requires transitive inference (Discourse Incrementation, Figure 5.5.). Moreover, when the discourse contains multiple referents, the model can bind the appropriate attributes to the correct referents (Two Referents, Three Referents, Figure 5.5.). It can even do so when these referents share one or more attributes (Problem of Two, Problem of Two Multiple Attributes, Problem of Two Category Inference, Figure 5.6.), thereby forming a potential solution to the Problem of Two. Not only is the model capable of behaviour at which symbolic models excel, it can also deal with subtasks connectionist systems are known to be good at. The perfect accuracy on the Attribute Inference and Category Inference subtasks show that it can deal with graded attribute information. It can infer that a referent who wears a lab coat is (more likely) a scientist, and vice versa.

The model also showcased behaviour that requires combining symbolic and connectionist capacities. First, while it can use the graded information stored in its semantic network to do inference, it can overrule the correlations that exist between attributes (Correlation Violation). Secondly, it can store multiple similar attributes of two referents - while keeping them distinct - and then infer their category (Problem of Two Category Inference).

3.2. The lesion model results support several theoretical predictions on the strengths and weaknesses of symbolism and connectionism

The two types of lesioned bundle memory models (symbolic vs. connectionist/Bayesian) struggled with two different subsets of the tasks. The Overly Symbolic model is great at tasks that do not involve graded information and can easily represent multiple referents correctly, including the Problem of Two subtasks that did not involve graded inference.

The two Bayesian models were good at modelling tasks where graded information is required, but had trouble with all other subtasks, even the simple One Referent subtask. Their bad performance on these simpler subtasks seems to have been largely because the attributes that were encountered in the questions would interfere with the information presented in the declarative sentences. This suggests that the answering of questions already requires keeping two similar referents separate: the referent the question is about and who is introduced earlier, and a temporary placeholder referent who is the subject of the question. Answering questions demands a comparison between these two. It is possible that since the Bayesian models had no mechanism to distinguish the two, they could not do the comparison necessary for answering the question.

Finally, none of the lesioned models were capable of solving the subtasks that required both symbolic bundle memory and connectionist graded inference (Six Questions, Correlation Violation and the Problem of Two Category Inference subtasks). Besides supporting theoretical predictions, the lesion model results also show which parts of the bundle memory model are necessary for which subtasks.

3.3. Connectionist systems with recurrence (RNNs and LSTMs) can also perform all subtasks

The RNN and LSTM performance of our best RNN and LSTM models (see Figure A1.) defied our expectations. They managed to perform reasonably well (RNN) or perfectly (LSTM) across all subtasks. They were able to generalise to new unseen combinations of attributes and were able to do so with a lower number of units (1000 units in the RNN/LSTM layer) than the number of units required for a code using disentangled features (707 possible attributes x 3 possible referents, see Discussion section on computational efficiency). One reason for why the RNNs performed worse could be the number of parameters in the model. A single LSTM unit has more free parameters than an RNN unit. Perhaps with more units, RNNs could reach comparable accuracy. The positive correlation between the number of units and accuracy does support this idea (see Figure A1.). Taken together, the success of both types of recurrent models suggests that they are able to learn to have the same capacities as the bundle memory model. The potentially far-reaching implications of these results will be discussed in the Discussion.

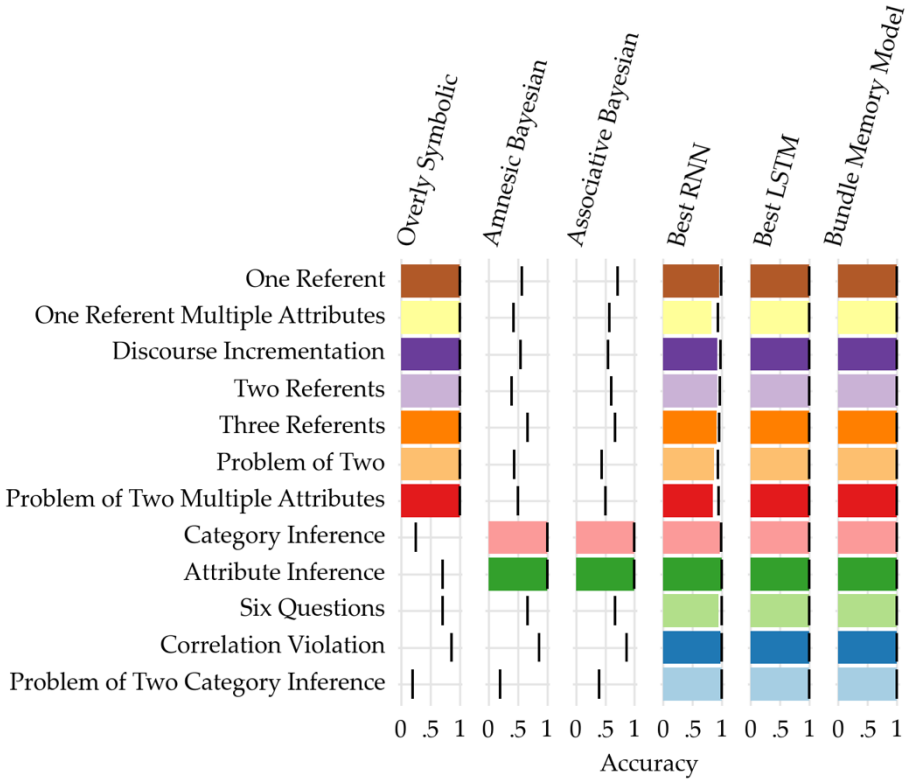
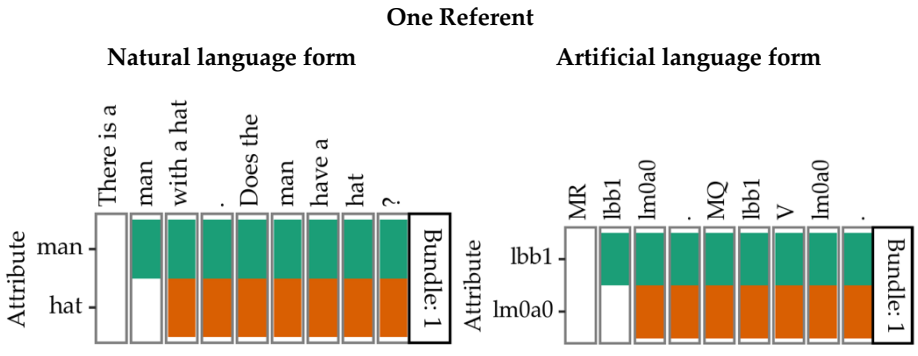


Figure 4. Accuracies of all models for all subtasks on the test set.
Coloured bars indicate lowest accuracy of all conditions within a subtask. This is the primary metric because many of the conditions are simple filler and control conditions. Thus, even if a model has a low accuracy rating on only one of the conditions, then the model does not have the capacity the subtask is meant to test. Vertical bars indicate mean accuracy over all conditions within a subtask, including control and filler conditions.



Discourse Incrementation

Attribute
Smith
man
hat
bag
Mr.
Smith
is a
man
.
The
man
has a
hat
.
The person
with the hat
has a
bag
.
Does
Smith
have a
bag
?
Bundle: 1

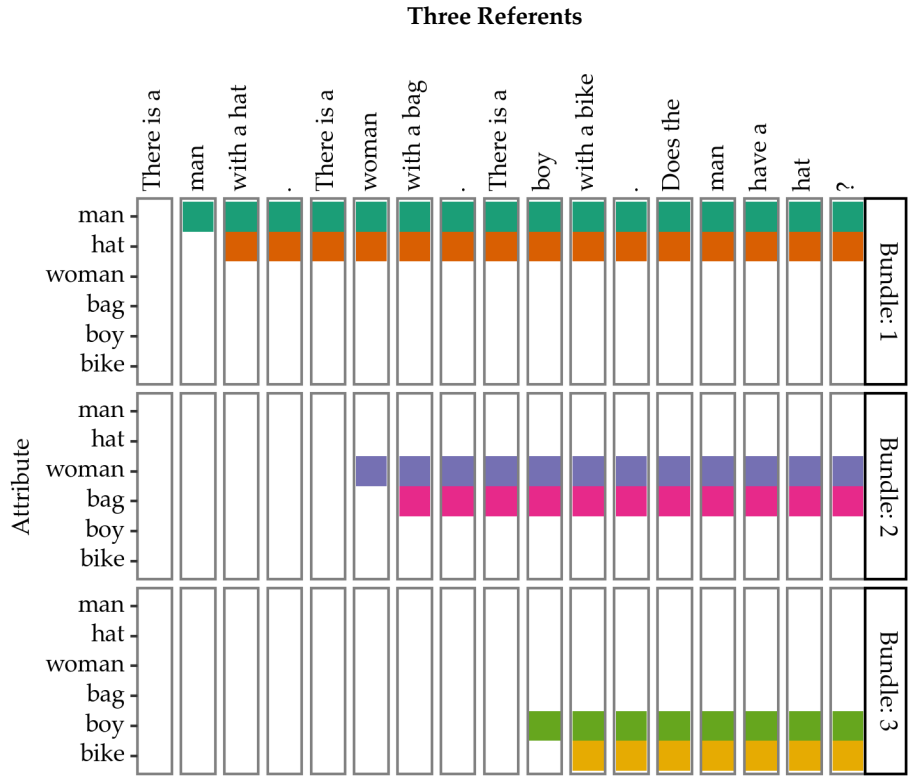


Figure 5.5. Activation state of bundle memory module as a function of time.

The One Referent example is a simple example, both in human-readable- and in model input form. The model input consists of control input (MR, MQ, V, .) and content words (lbb1, ln3a1). The Discourse Incrementation example shows how the bundle memory model incrementally updates a single bundle memory module as information comes in. The Three Referents example shows how the model stores the correct attributes of multiple referents to the appropriate bundle.

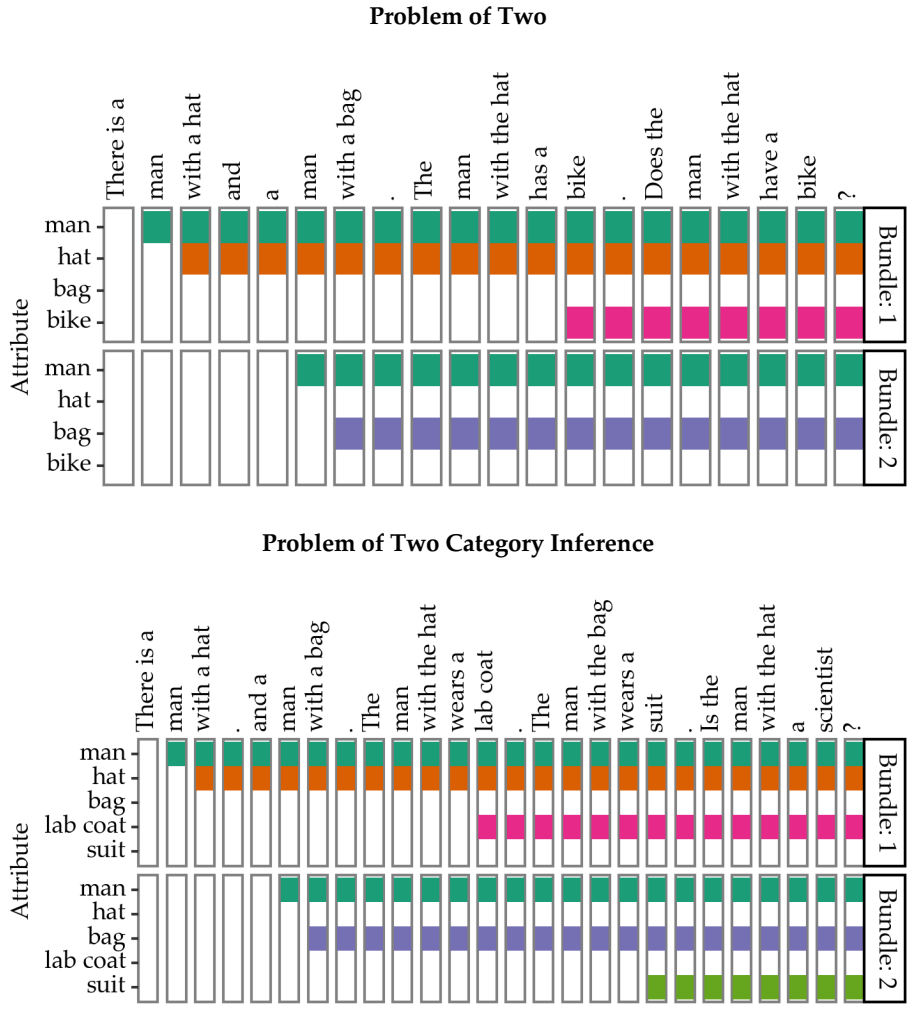


Figure 5.6. Activation state of the bundle memory module as a function of time.

The Problem of Two example shows that the model binds the attribute of the third sentence (bike) only to the correct referent (man with hat). The Problem of Two Category Inference example shows multiple additional attributes (lab coat/suit) being added to the correct referent (man with hat/man with bag).

4. Discussion

4.1. Summary of results

We propose a new computational model for reference comprehension: the bundle memory model. We showed that this model is capable of answering semantic questions involving one or more referents. This included discourses that required binding arbitrary combinations of attributes (binding problem) and those that contained multiple similar referents (Problem of Two) as well as questions in which graded information was required. With our lesioned models we have shown that the success of the bundle memory model is dependent on two complementary submodules: a semantic connectionist/Bayesian network capable of dealing with graded information; and a discrete, symbolic bundle memory and control module capable of binding arbitrary attributes and correctly distinguishing multiple referents with similar attributes. Finally, against prior expectations that were based on both theoretical and empirical grounds (Lake & Baroni, 2018; Loula et al., 2018; Sorodoc et al., 2020; van der Velde & de Kamps, 2006), we show that LSTMs and RNNs are capable of producing the same referential behaviour as our bundle memory model.

4.2. The bundle memory model can perform rudimentary reference comprehension

By complementing a connectionist semantic network with a symbolic bundle memory module and control network, we have created a model capable of rudimentary reference comprehension. The connectionist semantic network excels at storing the gradual, similarity-based and probabilistic information about attributes and their relations in long term memory, but faces difficulties with the binding problem and Problem of Two, especially when correlations are violated or entirely new referents are produced. Symbolic models will have difficulty dealing with the gradual, similarity-based, and probabilistic information about attributes, but offer a good solution to the problems facing connectionism in the form of dynamic variable binding. Like other similar models and hypotheses proposed in other fields of cognitive science (Baggio & Hagoort, 2011; Cowan, 2001; Eliasmith et al., 2012; Graves et al., 2016; Hadley & Hayward, 1997; Hayworth, 2012; Hayworth & Marblestone, 2018; Kahneman et al., 1992; Kriete et al., 2013; Lades et al., 1993; Manohar et al., 2019; Meeter & Murre, 2005; Schmidhuber, 1992; Smolensky, 1990; Teyler & DiScenna, 1986), bundle memory combines the best of both worlds.

4.3. Unlike some connectionist systems, bundle memory is computationally efficient

The computational efficiency of reference comprehension should be an important concern for all computational modelers of cognition. It is especially concerning for connectionist and Bayesian modellers, who often create networks whose relatively inflexible anatomy is supposed to contain at least as many states as there are possibilities - in our case, it should contain as many states as there are possible referents. The networks then compute by (the more flexible) spreading of activation over weights and nodes in the network such that the appropriate possible state (=referent) becomes active. This state must then be read out by subsequent (motor) regions to produce appropriate behaviour. Unfortunately, as we shall see, the compositionality and productivity of reference threatens the computational efficiency of such models.

The compositionality and productivity of reference is a statement akin to the compositionality and productivity of language (besides reference there exists compositionality of phonemes, morphemes, syntactic structures, etc.) and basically states that all combinations of object attributes are possible - though not equally probable and, realistically, constrained by the working memory capacity limit. The problem for typical connectionist and Bayesian models is that the set of possible referents is so large that storing all referents as possible states in the network's anatomy is computationally inefficient.

Similar arguments have been put forward many times before (Fodor & Pylyshyn, 1988; Jackendoff, 2003; Marcus, 2001; van der Velde & de Kamps, 2015). What complicates this discussion is that the computational efficiency of a connectionist/Bayesian model depends on the type of coding scheme they employ (Thorpe, 1989). In the following we will discuss the computational efficiency and capacities of four possible coding schemes, two of which remain popular to this day and one alternative solution that is similar to our bundle memory mechanism. We will assume a maximum working memory capacity of three conjunctions of three attributes each (Luck & Vogel, 1997; Oberauer & Eichenberger, 2013). Thus, we assume that under optimal conditions, people are capable of representing sentences containing three referents (consisting of 3 attributes each), e.g. "A strong hungry astronaut saw a sad blind clown eat a delicious salted tiger." Moreover, we assume that people can distinguish between different such representable sentences.

The average 20-year-old native speaker of American-English knows roughly 42,000 different lemmas (Brysbaert et al., 2016), i.e. uninflected words minus proper nouns (=names of people and places). The number of lemmas does not equal the number of attributes, given semantic overlap between words and the existence of function words, so we assume a more conservative number of 10^4 different possible attributes. For simplicity, we also assume no complex interactions between words, e.g., the order of attributes does not matter and there are no gestalt phenomena that are more than the sum of the parts.

The number of possible referents $\boldsymbol{r}$ (with exactly 3 attributes) you can create with these assumptions is $\binom{10^4}{3} \approx 10^{11}$. The number of representable sentences $\boldsymbol{s}$ (of the form "x saw y ate z") you can make with 10^{11} possible referents is roughly $(10^{11})^3 \approx 10^{33}$. For comparison, it has been estimated that there are roughly 10^{14} synapses in the brain (Drachman, 2005). Note that this is still a lower bound of the number of representable sentences, since we do not include referents with only 1, 2 or with 4+ attributes, nor do we include sentences with different syntactic structures or verb phrases.

The first coding scheme is called a conjunction code, or a pure local code (Thorpe, 1989). This code uses a distinct node in the network for every representable sentence (=conjunction of referents, where each referent is a conjunction of attributes), i.e. the number of nodes $\boldsymbol{n} = \boldsymbol{s} \approx 10^{33}$. Standard Bayesian models require this many nodes to express the joint distribution of all representable sentences. This is not a very memory efficient solution, and therefore unlikely to be the solution employed by the brain given that it contains only about 10^{14} synapses (Drachman, 2005).

The following two codes each employ a different strategy to reduce the number of nodes required. Each has its up- and downsides.

The second coding scheme uses so-called disentangled attributes. Here, each attribute gets its own node (=10^4 nodes). A referent is then represented by the simultaneous activation of its attributes. This allows straightforward binding of any number of attributes. Multiple referents are represented through a positional code. This means the three referents of our example have to be represented in three separate copies of the attributes, i.e., $\boldsymbol{n} = 10^4 \times 3 \approx 10^4$. This coding scheme is much more memory efficient than the conjunction code. However, the positional code still requires a lot of seemingly unnecessary duplication, especially when we consider that multiple 1-attribute referents each require a separate copy of the attributes. Thus, a sentence with 7 such items, e.g. "A barber, a butcher, an acrobat, a DJ, a hockeyer, a knight and a scientist went skiing together." requires 7 copies of the attributes in this code, i.e. $\boldsymbol{n} = 10^4 \times 7 \approx 10^5$. Not only is this still an inefficient solution, but it also introduces the problem of how the brain keeps these 7+ copies of the semantic network semantically similar (Hayworth, 2012) - especially over longer time scales, when meanings change or new concepts are learned.

The third coding scheme is called a distributed code and is likely the most popular coding scheme in neuroscience (Thorpe, 1989). This uses a population code where there is no 1:1 mapping between nodes and attributes. Instead, in a distributed code most of the nodes provide some information about each of the attributes. A particular referent is then encoded by a particular combination of nodes (or equivalently by a particular direction of the vector containing all the node activations).

The advantage of this code is that it can potentially compress the information more densely by utilising the sparsity of the input: it can use fewer nodes by letting individual nodes represent a superposition of multiple attributes that are unlikely to appear together (Elhage et al., 2022). Theoretically this strategy makes it possible to encode $\exp(n)$ items with

$\boldsymbol{n}$ nodes (Elhage et al., 2022). That is, the 10^{33} representable sentences could be encoded in only $\ln(10^{33}) \approx 76$ nodes.

However, this assumes sparsity in the combinations of attributes or referents. It is true that our linguistic input is sparse: not all combinations are meaningful and present in our input. Nevertheless, syntactically, all combinations are allowed and people are in principle capable of representing any of them. That is, even though syntactically sound, meaningless sentences might be more difficult to remember, it should not be completely impossible. As an existence proof of this statement consider the sentence: "A triangular round square saw a green orange penguin eat a featherless flying penguin." This sentence contains referents with semantically incoherent attributes, but can still be processed and remembered. There is not a total loss of meaning either, e.g. we can still answer the question of who got eaten. Therefore, the content of this sentence must somehow be representable. And since our coding scheme has to encompass all representable sentences, without any sparsity to make use of, the system must be capable of representing the entire space of possible combinations. A distributed code can therefore not compress information to a smaller subspace, meaning that it should still require an equal number of nodes as the disentangled code, i.e., $\boldsymbol{n} \approx 10^5$.

In short, the above three coding schemes each have trouble dealing with the compositionality of reference. In our opinion, the reason is because they assume that all of the representable sentences must be encoded in the slowly-changing network structure from the start. In contrast, our bundle memory model only stores a single copy of the basic attributes, and then dynamically creates referents (and sentences) on the fly. It does so by quickly adjusting small parts of the network structure in the bundle memory module only, while leaving the larger semantic network intact. As a result, it only requires nodes to store the attributes, i.e. $n = 10^4$, plus some additional bundle nodes: for (flat) sentences with three referents, three bundle nodes are required, i.e. $n \approx 10^4$. This makes it the most computationally efficient solution of the ones considered here.

Importantly, the above considerations do not entail that the above coding schemes are impossible. After all, with a sufficient number of additional nodes, it should be possible to implement a system like this. In fact, we are indifferent to which coding scheme the connectionist semantic network of our bundle memory model uses - we expect the brain uses an eclectic combination of codes. Our point here is that we should not only consider slowly-changing coding schemes, when dynamic solutions can be more computationally efficient. Therefore, the above should not be seen as a criticism of connectionism/Bayesianism as a whole, but merely a showcase of the computational efficiency issues with popular pure connectionist/Bayesian coding schemes when it comes to the compositionality of reference. Instead, pure connectionist/Bayesian systems should be supplemented with bundle memory-like inductive biases for dynamic variable binding that enable on-the-fly creation of referents.

4.4. The bundle memory model is biologically plausible

One common complaint against symbolic mechanisms is that there have been few, if any, biologically plausible implementations of such mechanisms (Eliasmith, 2013). Here we wish to justify the biological plausibility of our symbolic bundle memory module.

Our first order of business is to discuss a major assumption behind the bundle memory model: namely that the brain separates semantic memory and working memory. That is, it can store and retrieve items (e.g., referents) that are composed of semantic attributes stored in long-term memory into- and from separate working memory buffers. While popular in the past (Atkinson & Shiffrin, 1968; Malmberg et al., 2019), this view went out of fashion when neuroscientific studies did not discover a clear separation between long-term memory and working memory in the neocortex: cortical areas that are associated with long-term memory are also active during working memory maintenance and retrieval (Christophel et al., 2017; Lewis-Peacock & Postle, 2008). However, absence of evidence is not proof of an absence. While it seems true that areas that store long-term memory representations remain actively involved during working memory, this does not mean that there are no separate working memory buffers in the brain anywhere. In fact, our bundle memory model would predict such involvement, since the bundle memory module only contains semantically vacuous representations and therefore requires the semantic network for filling in all of the details. Moreover, several areas that are implicated in working memory - like the prefrontal cortex, parietal cortex, and ACC, but also the hippocampus, and basal ganglia (Chai et al., 2018; Chatham & Badre, 2015; Moscovitch et al., 2016; Vilberg & Rugg, 2008) - are connected to, but still largely anatomically distinct from the long-term memory representations found in sensorimotor cortex. Thus, the assumption of separate semantic long-term memory and working memory should still be in line with the empirical neurobiological facts.

With that out of the way, we will go over the main mechanistic requirements that are assumed by our bundle memory model and for each propose what their biologically plausible implementation could be.

Both the semantic network and the control system can be implemented in fairly standard connectionist models. In principle, the semantic network can be implemented as a feedforward neural network; the control system in its current form requires some memory to remember its current state and could therefore be implemented as a recurrent neural network.

The bundle memory module has more requirements, yet still nothing exotic. Most of the heavy lifting within the module is done by the wiring of the circuitry. Importantly, variants of the bundle memory module have previously already been implemented as neurobiologically plausible rate-coding model and spiking neural network (Manohar et al., 2019; Fiebig et al., 2020).

The first requirement of the module is that it needs a binding mechanism that can establish rapid associations between bundle nodes and attribute nodes. Such rapid binding can for instance be achieved by fast Hebbian long-term potentiation of the synapses between

a bundle node and an attribute node. This form of learning can be induced during between 0.5 to 5 seconds and remain for hours (Gerstner, 2011). Hebbian long-term potentiation was first observed in the hippocampus but is thought to be present throughout the brain (Gerstner, 2011; Malenka & Bear, 2004).

Second, the bundle nodes require anatomical connections to all attributes - since they must be able to bind arbitrary attributes together. It is possible that some areas in the brain contain single neurons that have connections to all attributes. However, given the distributed nature of the brain, it is more likely that each neuron is only connected to a subset of attributes. Multiple such neurons could be connected to one another, jointly forming a subpopulation that acts as a cohesive bundle node that - as a population - can bind to all attributes. The hippocampus and basal ganglia are thought to in principle possess the neural circuitry needed for learning of associations between any arbitrary attributes (Duff et al., 2020), but there is evidence that other multimodal areas in the neocortex (prefrontal/anterior temporal/parietal cortex) do too (Sharon et al., 2011).

Third, the bundle memory module requires the capacity of selectively strengthening only the connections between a bundle node and the appropriate attribute node(s). This is needed because a single bundle node is connected to all (or many) attribute nodes, but should only bind the appropriate attributes. This can be implemented in biological neurons through synapse-selective or compartmentalised plasticity in which learning only takes place in one or only a few of the many connections a neuron may have. Such synapse-selective plasticity is a known feature of certain neurons, for instance those found in area CA1 of the hippocampus (J. Lisman et al., 2012; Nimchinsky et al., 2002).

Fourth, the bundle memory module must form a loop with the semantic network, since the module should be able to not only store but also retrieve information. Furthermore, the incoming connections from the attribute nodes to the bundle node and the outgoing connections back to the attribute nodes must be aligned in some way. This is necessary to ensure that the attribute that gets stored in bundle memory is roughly the same one that is later retrieved. The hippocampus and the basal ganglia are both known to have this type of connectivity with the cortical semantic network (Haber, 2016; Milardi et al., 2019; Rolls, 2017; Teyler & DiScenna, 1986). For the hippocampus, it is by now well established that it can drive the cortex to replay memories of previously stored information (Ólafsdóttir et al., 2018). The basal ganglia and sensorimotor cortex are both topographically organised and connected together so that parallel streams with the same function pass through the basal ganglia to and from cortex, forming a cortico-basal ganglia loop (Alexander & Crutcher, 1990; Haber, 2016). Additionally, other areas like the prefrontal and parietal cortex are likely to have similarly recurrent and aligned connectivity with the sensorimotor cortices in order to be able to fulfil their purported roles in working memory and attentional control (Parks & Madden, 2013).

Finally, in order to ensure that the attributes are stored and retrieved only by a single bundle node, the model assumes a sparsity constraint: only a single bundle node at a time

should spike fast enough to trigger learning or retrieval. This can be implemented through lateral inhibition of other bundle nodes, which is known to result in sparse winner-takes-all dynamics (Coultrip et al., 1992).

The use of lateral inhibition to decide between referents is complicated by the fact that referents can be completely novel and only distinguished by arbitrary attributes (Problem of Two). It takes several minutes to hours to form a new synapse - and new referents can be introduced and compete with one another at timescales far shorter than that. Thus, winner-takes-all mechanisms between referents likely require already established lateral and inhibitory connections. The Problem of Two makes a solution based on lateral inhibition at the level of attributes impossible, since modulating the attributes will not allow distinguishing between two similar referents, i.e. two combinations of attributes where one or more attributes are shared. Finally, as we have seen, competition at the level of referents without a bundle-like mechanism, e.g. through a conjunction-code, is computationally inefficient. Thus, competition is best achieved by lateral inhibition between bundles. Their distance from the semantic network means bundle nodes are like empty vessels that can be filled with whatever arbitrary combination of attributes is desired: the lateral inhibitory connections between bundle nodes are already set up, but the content of what these neurons fire for is bound to the empty vessel in a dynamic manner. The distance makes these bundle nodes symbol- or variable-like, and with their pre-existing lateral connections you can have winner-takes-all mechanisms that decide which referent - out of several alternatives - is meant, even when it has only just been introduced in the discourse. Sparse firing through lateral inhibition are well known properties of both the hippocampus (Cayco-Gajic & Silver, 2019; Espinoza et al., 2018) and the basal ganglia (Groves, 1983; Mink, 1996) but for recent counterevidence see (Park et al., 2020). However, lateral inhibition can be found throughout the brain (Rolls, 2021).

Taken together, we conclude that all of the ingredients for the bundle memory model are available in the brain. This makes the model biologically plausible - in contrast to similar models that make use of mathematical operations such as circular convolution or tensor products without clear biological implementation (Gayler, 2004; Smolensky, 1990). Whether the ingredients are also composed together in the way we described above - that is, whether the model is also biologically real - is an empirical question left for the future. Even if it turns out that this exact neurobiological implementation is not the way the brain performs reference comprehension, we still expect the real implementation to roughly correspond and obey the computational operations highlighted by our model.

4.5. Limitations and future directions of bundle memory model

The bundle memory thus provides a computationally efficient and neurobiologically plausible solution to the complete set of S3R problems, and thereby of reference comprehension. Yet, there remain some important limitations, primarily stemming from the

simplifying assumption that sentences are without hierarchical structure. This assumption is surely false - language contains considerable hierarchical structure (Everaert et al., 2015). This includes referent chunking, e.g. "Bob and Rob" → "They"; nested referents, e.g. "Bob's father"; and transitive relations "Bob bites Rob". Constructions such as these will lead to problems for the flat conjunctive bundle mechanism.

However, there are good reasons to be optimistic. While our model does not fully close the wide gap between linguistic theory and neuroscience, it does make it considerably narrower. The main reason is our model's capacity to store bundles into discrete "registers" through controlled addressable read-write memory. As discussed in the introduction, this capacity enables dynamic variable binding - ie. the ad-hoc creation of discrete symbols that can take arbitrary values - which is considered a key requirement for modelling compositional linguistic representations (Fodor & Pylyshyn, 1988; Marcus, 2001). But it also enables other computational mechanisms that computers use to parse the hierarchical structure found in programming languages.

One problem that could be solved by the use of bundles is how to store referents that are chunked together (e.g. "Bob and Rob" → "They"). Although phrases like "Bob and Rob" are literally conjunctions, they are problematic for our flat conjunction mechanism. This is because you can refer back to both the collective "they", and to the individuals "Bob" or "Rob", whereas our current bundle memory mechanism will only be able to retrieve the collective. Importantly, it is not computationally efficient to store all possible combinations in a large network: because we can construct phrases containing arbitrary individuals, this likely involves a form of dynamic variable binding. Ideally then, you need a way to dynamically create a composite representation that contains the individuals - a data structure. One way the brain could do so is to use appropriately constructed recurrent connections to create hierarchical bundles, i.e. bundles that can bind and thereby point to other bundles (as well as attributes). Thus, the brain would set up a bundle that stands for the conjunction of Bob and Rob, which is bound to two bundles: one pointing to Bob, the other to Rob, thereby dynamically and efficiently storing these chunked referents.

A second important problem - that of transitive relations between two referents - can be solved in a similar way. Transitive relations such as "Bob bites Rob" are challenging for flat bundle mechanisms because these cannot represent who did what to whom. These constructions are also thought to require dynamic variable binding to bind arbitrary attributes to predicates (also known as role-filler independence) (Hadley, 2009; Holyoak & Hummel, 2000; Marcus, 2001). It has been suggested that 2-place predicates like Bite(x, y) can be parsed as two separate 1-place predicate bundles, Bite (x) and Is Bitten(y) (Doumas et al., 2008; Seuren, 2009). The brain could then store these in separate bundles. But this is not sufficient to represent who bit who in sentences like "Bob bites Rob who bites Job". To represent these, the two 1-place bundles would also need to be bound to one another through a hierarchical bundle representing the composite 2-place predicate - disambiguating the first and the second

bite-event by storing them in separate hierarchical bundles. Transitive relations containing arbitrary numbers of arguments, so-called N-place predicates, can be constructed using larger numbers of hierarchical bundles.

Finally, the last problem involving hierarchy is that of nested referents (e.g. the brother of the father of the girlfriend of Bob). For dealing with such hierarchies, computers often use a stack, a set of registers that are accessed in the reverse order items are stored. Stacks can be used to store temporary results of functions - or in the case of natural language, predicates. For instance, if the girlfriend of Bob is Lisa, then you can first parse and store "the girlfriend of Bob" as "Lisa". Then, "the father of the girlfriend of Bob" becomes "the father of Lisa", and so on. The brain could implement a stack by recurrently connecting multiple bundles together so that they form a sequence. By incrementally storing the temporary results of predicates on such a stack, the brain might sequentially store and retrieve these bundles to sequentially parse arbitrarily large nested referents (including recursive ones, e.g. the father of the father … of the father of Bob).

The above mechanisms for dealing with hierarchy are not too difficult to implement themselves. The difficulty lies mostly in the construction of a sufficiently complex control system that can correctly decide which things to bind to which bundles. Thus, modelling referent chunking, nested referents and transitive relations remains an open problem. However, in contrast to models that propose to solve the Problem of Two by treating referents as different attractor states in the dynamic state space of their model (Fitz et al., 2020), bundle memory offers clear ways to control the dynamically created referents. This is because the bundle nodes are like static registers that can be controlled by pre-existing anatomical connections from control areas. Although bundle memory can, of course, also be described as a dynamical system with attractor states, our account is more explicit about how a control system can choose to distinguish, store or retrieve specific dynamically created referents using bundle nodes. Moreover, the introduction of the biologically plausible bundle memory module enables the formation of testable hypotheses of how the brain implements linguistic hierarchical structure, which is more difficult for models that view hierarchical structure exclusively as emerging properties of highly complex dynamical systems.

Bundle memory also offers some advantages compared to previous modelling attempts that assume a fixed number of hierarchical relations (e.g., Smolensky, 1990; van der Velde & de Kamps, 2006, for argumentation see Eliasmith, 2013; Hadley, 2009; Martin & Doumas, 2017). The aforementioned mechanisms promise a way of representing hierarchical relations that generalises to arbitrarily complicated sentences. This is because the dynamic variable binding mechanisms proposed above are independent of the depth of the syntactic tree, or of the length of the discourse - and only assume sufficient memory capacity. Of course, all of this assumes that the above promises can eventually be fulfilled and thus depends on the success of future modelling efforts.

Another important limitation that subsequent work should address is the dependence of the control model on discrete and external information provided by definiteness ("the" versus "a"). Ideally, the model should be able to (learn to) decide - using its own internal goals - whether to bind an attribute to a new referent or to an existing one. Moreover, definiteness often does not provide perfect information about whether a referent is new or previously introduced (Gundel et al., 1993). This can likely be achieved by replacing our discrete control system with a continuous recurrent neural network that has learned how to transform the graded definiteness signal into the operations controlling the bundle memory module.

4.6. Symbolic and connectionist mechanisms complement each other

We have shown that our lesioned models (symbolic and Bayesian models) - though capable of solving a subset of our referential test suite - are unable to exhibit the full range of correct referential behaviour. We caution against generalising the failure of our purely symbolic and Bayesian models to the larger class of symbolic or Bayesian models. In fact, sufficiently complicated and properly wired symbolic or Bayesian models will surely be able to do the job: for instance, those that implement our bundle memory model. Yet, our results do corroborate that some of the models that were - or still are - popular in cognitive science, lack important mechanisms needed for computationally efficient solutions to reference comprehension. Interestingly, some Bayesians have recognized this fact and have started to add compositional mechanisms to Bayesian models (Bernardy et al., 2018; Goodman et al., 2014).

4.7. Are RNNs and LSTMs good models of reference comprehension?

Contrary to our own prior expectations - and those of an influential part of the literature (Fodor & Pylyshyn, 1988; Jackendoff, 2003; Marcus, 2001; Puebla et al., 2020; Sorodoc et al., 2020; van der Velde & de Kamps, 2015) - our best RNN and LSTM models succeeded in learning to perform rudimentary reference comprehension. It has long been held that recurrent neural networks could not i) answer semantic questions, ii) resolve referential long-distance dependencies, iii) bind arbitrary attributes together, iv) deal with violations of learned correlations, or v) separate two similar referents. The success of these models on our dataset (Figure 4.) challenges these ideas. There are several possible interpretations of this finding.

One possibility is that the RNN and LSTM models have found a way to compress the information in our test suite into a distributed code in order to successfully answer our prepared questions, but do no more than that. In this case, these models still fail to find a computationally efficient solution to the understanding of larger discourses in spite of their success in the set of sentences we used: though a dataset containing a million sentences and

roughly 700 content words is not small by any means; it is still considerably smaller than the number of sentences and content words found in natural language. Evidence for this is that the trained RNNs/LSTMs do not always give correct answers to queries that are different from those found in our training and test sets. For instance, it struggles answering correctly when we make the sequence of sentences longer than it was trained on. If true, this would predict that these networks would still struggle with discourses containing rare occurrences of attribute/referent combinations or with larger discourses (see Section 4.3. on computational efficiency). This can be more rigorously tested by evaluating model performance on a corpus containing all possible referent combinations and by increasing the lengths of the discourses.

Yet, it is conceivable that the RNN and LSTM models have indeed successfully learned how to store and retrieve arbitrary bundles of attributes in a computationally efficient manner. If we take this possibility seriously, then there are two further possibilities. One is that the RNNs and LSTMs have learned a bundle memory-like mechanism which would allow them to dynamically create multiple referents, flexibly bind multiple arbitrary attributes to them, and store the correlations between these attributes and the larger categories to allow for inference. In this case, we should say that these successful models have learned how to implement symbols, in line with suggestions of implementational connectionism (Fodor and Pylyshyn, 1988; Marcus, 2001; Pinker and Jackendoff, 2005). However, there is little evidence for this: when we investigated the internal states of our model, we did not find any units (or collections of units) that indicate the representation of a particular referent mentioned in the discourse. It is nevertheless possible that the models implement a bundle memory mechanism in a more abstract, distributed manner not immediately obvious through visual inspection of the model's states. Further research into how these models work internally could help determine how they pass our test suite, thereby potentially answering such questions.

A second possibility is that the successful RNNs and LSTMs have found a completely different way to solve the binding problem and Problem of Two in a computationally efficient manner. If so, then these models provide evidence against the need for symbols in reference comprehension, i.e., eliminative connectionism. This would not necessarily mean that humans are not explicitly symbolic, but it would mean they could be: either RNNs and LSTMs can learn to comprehend language in a different way from humans, i.e., reference comprehension is multiple realisable (Putnam, 1967), or perhaps humans do not explicitly represent symbols either. If so, a careful study of the model's internal states could reveal a non-symbolic mechanism that could serve as an alternative hypothesis of how humans perform reference comprehension.

In any case, the fact that these models were able to successfully answer semantic questions about complicated discourses is a surprising and interesting finding in itself. This is especially so given the fact that similar types of recurrent models trained on next word predictions often fail at anaphoric reference (Kocijan et al., 2023; Sorodoc et al., 2020). It is possible that the next-word prediction learning paradigm that is often employed to teach these

models, does not sufficiently challenge the model to learn the capacity for anaphoric reference that is necessary for true discourse semantics. Training an LSTM using the next-word prediction paradigm on our test suite could determine whether previous LSTM models failed because of this training regime.

4.8. Conclusion

We have introduced the bundle memory model for reference comprehension, combining symbolic and connectionist elements to solve complex linguistic challenges. We believe that hybrid models like ours offer a promising approach for creating computationally efficient and neurobiologically plausible models of reference comprehension, and possibly of language more generally. In addition to the primary modelling effort, we found RNNs and LSTMs to perform surprisingly well at reference comprehension, challenging conventional thinking about the limitations of such models. At this point in time, the jury is still out on whether bundle memory is needed for reference comprehension, or whether the potentially simpler recurrent neural networks can do the job just as well *and* in a biologically realistic manner. Nevertheless, it should be clear that both the introduction of the bundle memory model and the surprising success of recurrent neural networks open up exciting new research directions that could reveal how the brain comprehends language.

Appendix A

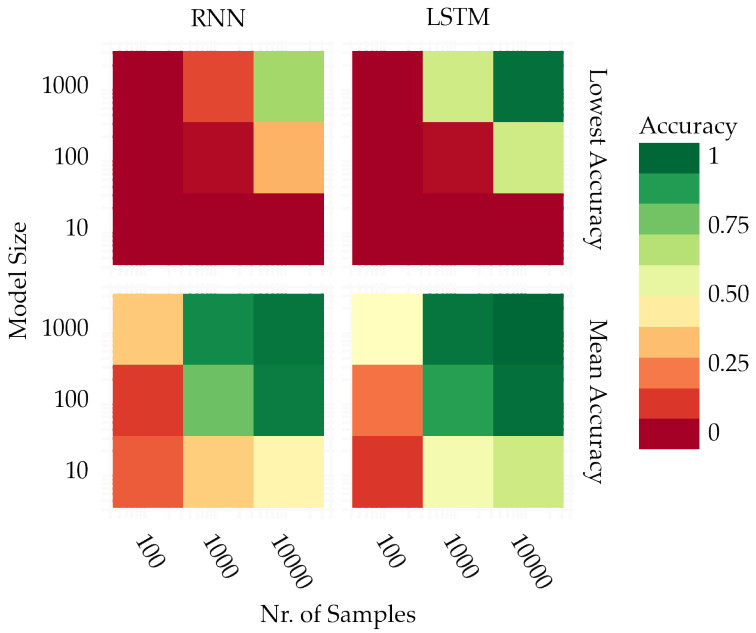


Figure A1. Lowest and mean test accuracy of RNN and LSTM models as a function of model size and number of samples.

Model size indicates the number of units both in the embedding layer and in the hidden RNN layer. With the number of samples, we mean the number of sentences we generated per test condition.